OUTDOOR SLAM USING STEREO VISION AND SIFT FEATURES

Herman Kloosterman (1239309)

30 October 2007

Internal supervisor:

Dr. B. de Boer

Artificial Intelligence,

Rijks*universiteit* Groningen,

The Netherlands

External supervisors:

Ir. E. den Breejen &

G. Dubbelman Msc.

TNO Defence, Security and Safety,

The Hague, the Netherlands

Keywords: SLAM, Loop closure, Stereo vision, SIFT

1 TABLE OF CONTENTS

2 ABSTRACT

Simultaneous localization and mapping (SLAM) is a central subject in the field of autonomous robotics. SLAM is the process of localizing a moving robot in an unknown environment. This calls for localizing the robot relative to the map. On the other hand, the map has to be constructed while the robot moves, which makes SLAM a "chicken-and-egg" problem. Furthermore, the localization and the map must both be derived from sensor readings, which contain noise.

In the frame of this study, a vision only SLAM implementation is achieved. Large-scale outdoor experiments are performed at TNO Defense, Security and Safety, in natural and urban terrain. The environment is perceived by sparse stereo vision using scale invariant image features (SIFT) as interest points. Successive stereo images are used to estimate the displacement of the robot, which causes a cumulative error. The localization error can be decreased when the robot is located at a point it has been before when the position of the robot was known more precise. Detecting such a point is called loop closure. To reduce the calculations needed for recognizing loops, feature selection can be used.

For the experiment, the conclusion is that on a route of 1100 meters, by neglecting the features of the ground plane, the search space can be reduced by a factor two without notably degrading loop closure performance. Preselecting the landmarks on the map based on their location, reduces the number of landmarks to be matched towards the end of the run by a factor 50. Using absolute matching distances, compared to relative matching distances, ensures more constant loop recognition when the number of landmarks on the map grows. Storing, not all the landmark information multiple times on local stored maps, but only once, on a global stored map, strongly reduces computational power and memory usage.

3 INTRODUCTION

Navigation in robotics is the skill in which a robot can steer itself from place to place. It is a central subject of scientific research in the field of autonomous robotics (Frese, 2006). The following questions are relevant: "Where am I?", "Where do I have to go?" and "How do I get there?" This research project is aiming at answering the "Where am I?" question. The robot must therefore be able to build a map of its environment (mapping) and in the same time track its own position on this map (localization). This is called SLAM, which is the abbreviation of simultaneous localization and mapping.

When investigating SLAM, a robot is placed in an unknown environment. The mapping task of the robot consists of locating the landmarks in its surroundings. A landmark can be an artificial or a natural object (or part of an object) that can be sensed by means of sensors. The position of the landmark, relative to the position of the robot can be estimated from these sensor readings. To derive the absolute position of the landmarks, the position of the robot has to be known. The position of the robot can only be deduced from its position relative to its surrounding landmarks. This is a "chicken-and-egg" problem.

As the robot moves, it can estimate its displacement with a limited accuracy. The positional uncertainty grows cumulatively with each movement. By measuring the environment, the robot gains information. The robot can use this information to reduce the uncertainty by comparing the current measurement to earlier measurements. Uncertainty reduction can particularly take place when a robot returns to a position and recognizes the landmarks. This is called a loop closure. The robot compares the landmarks it perceives at that moment, to landmarks on the map that were added when the position of the robot was known more precisely.

FIGURE 3-1, THE ROBOJEEP AT THE WAALSDORPERVLAKTE. THE DATA OF THE STEREO CAMERA, GPS, IMU AND ODOMETRY IS STORED. ONLY THE STEREO IMAGES ARE USED IN THIS EXPERIMENT AS INPUT FOR THE SLAM ALGORITHM. GPS DATA IS USED AS A REFERENCE.

3.1 PRACTICAL USE

SLAM techniques in general and vision based SLAM in particular, are so-called enabling techniques. When a robot can build a map of its surroundings and trace its own position on it automatically, this information can be crucial for tasks on a higher level like change detection, visualization, and so on (see Table 3-1 for examples). SLAM is therefore an interesting subject for the Defense, Security and Safety department of TNO in The Hague.

Change detection
• By detecting changes in its surroundings, for example an opened window, an autonomous surveillance robot can safeguard a building. • Soldiers on patrol can benefit from using a change detector. Changes in the environment might be a sign of a possible assault or a bomb hidden on the roadside.
Line of sight calculation
• By combining SLAM and dense stereo vision information from multiple soldiers who operate as a team, a 3D terrain model can be build. This model can be used for line of sight calculation, which can help soldiers to cover teammates. It can help the commander to gain information and localize his men. • Line of sight information can also help soldiers to find the best position to observe the enemy.
Visualization
• Seeing a map that is build and displayed in real time on a control panel enhances situation awareness for an operator who has to navigate a platform and observe the environment. This could also enable the operator to point at a previous point on the map, to make the platform retreat to that position semi-autonomously. • Experienced soldiers often have limited time to pass on environmental information to new arriving soldiers who will proceed their task. SLAM can be used to build a 3D terrain model. By means of this model combined with video layover the new soldiers can explore the environment beforehand using virtual reality imaging devices.

TABLE 3-1, TECHNIQUES THAT CAN BENEFIT FROM SLAM, WITH THEIR POSSIBLE USE

3.2 ROBOTIC VISION

In the setup, the environment of the robot is perceived by stereo vision. The angles to the landmarks, as seen from the robot, can be determined from the images. When using stereo cameras, the distance between the two cameras is fixed. Both cameras will perceive a landmark in the environment from a different position. Therefore, the position of the landmark in the left and the right image is slightly different, which is called disparity. From this disparity, the distance between the landmark and the stereo camera can be derived.

Image features such as lines, edges and corners can be used as natural occurring landmarks. To be able to compare the features from different images, they need to be detected and described first. The descriptor needs to be distinctive and robust. Robust in this context means that a landmark must have approximately the same constant description in various conditions. Examples of varying conditions are changes in light or differences in distance from where the landmark is perceived. A scale invariant feature transformation that is abbreviated by SIFT (Lowe, 1999) is used in this experiment, because the landmarks are perceived from various distances (see Figure 3-2).

3.3 Ego-motion estimation

The route of the robot can be calculated from its initial position and the transitions of the robot between the time steps. This process is known as ego-motion. The transitions of a robot that has the kinematics of a car can be estimated by measuring the wheel rotations together with the steering angle. Due to wheel slippage and other unmodeled factors, the position estimation error can grow quickly. This is especially the case when the robot has to drive trough irregular natural terrain. Although it is still used in most cases, odometry does not provide a good method to estimate the position over time. This is a shortcoming of traditional SLAM approaches

In this experiment, no odometry, IMU or GPS is utilized for ego-motion estimation. The applied ego-motion estimation approach is solely based on stereo vision (Van der Mark, W.; Gavrila, D. M., 2006). The environment is assumed static so if there are changes in the captured images these are due to changes in the pose of the stereo camera. The ego-motion is estimated by comparing the SIFT features of consecutive stereo frames, which can then also be used for mapping.

3.4 Vision based SLAM

The localization and the mapping of the SLAM algorithm in this experiment are both derived from the images of a stereo camera. Each time step the descriptor and the width, the height and the depth of the SIFT features compared to the camera are calculated from the stereo image. The location of the robot is derived from the displacement of the SIFT features in each time step.

Figure 3-2, an example of SIFT features on landmarks/objects (Lowe, 1999)

A particle filter will be used to solve the SLAM problem. The position of the robot will therefore be represented by multiple particles. If the position of the robot is known at the start of the route, all the particles will start on the same position on the map. When the robot moves, the particles are translated by the estimated displacement of the robot. Random normal distributed noise is added during translation.

Imagine that each particle represents the true pose of the robot, then for each particle, a local map can be made. The positions of the features on the map are calculated by transforming the position of the features as seen from the robot to x, y and z values in map coordinates. If the pose each time step is known, which is the route of the robot, a large map can be build by merging smaller maps that are perceived temporarily. Features on local maps that represent the same landmark as derived from their position and descriptor can be merged by using a separate Kalman filter for each landmark. This will be discussed in chapter 5.

The assumption that the estimated routes of all particles are the true route of the robot does not hold. Some are better and others are worse because of the error in the ego-motion estimation and the random added noise. The particles can be weighted by how well the current perceived features fit to the expected features, based on the pose of the particle and the map of the particle build so far. A particle that is near the true route will more likely expect to see features that match the current perceived features than a particle that did not estimate the true displacement of the robot so well. This is especially the case during a loop closure, when the robot is at a place it visited before during the route. By weighted redrawing of the particles, when the weights of the particles differ too much or near a loop closer, the better particles are more probable to survive, to the disadvantage of the worse particles.

3.5 Experiments

The experiments are conducted outdoor. The first route lies in a military test terrain and the second route lies in an urban area (see Figure 3-3 at page 12). The natural terrain of the first route is rough so a four-wheel drive car is needed to drive here. The Robojeep at TNO will be used as a test robot (Figure 3-1 at page 6). This is a commercially available "Jeep Wrangler 4.0i", modified to drive autonomously. The Robojeep has a stereo camera mounted at the front and a GPS receiver mounted on top, used for ground truth.

For the experiment, the jeep is driven manually. The stereo imagery and GPS data are stored on a computer for offline processing. For use of the dataset in future experiments, the steering angle, the wheel rotation speed and the readings of the Inertial Measurement Unit (IMU) are stored additionally. The IMU contains an electronic compass that measures the heading. The IMU also measures the orientation of the car compared to the earth's gravity. Ultrasonic range finders and a laser range finder are also available on the Robojeep, but these are not logged.

SLAM based on vision is tested for closing large loops. The research questions that will be answered are show in Table 3-2. The beacons will be naturally occurring landmarks. The distances from the robot to the landmarks are calculated by stereopsis. Optical features are used to distinguish between the landmarks by using SIFT feature descriptors. As a control measurement, artificial marks on the ground, aerial maps and GPS will be used.

Research questions
Is it possible to use vision only SLAM in a large-scale outdoor environment?
What kind of landmarks can be used best and how can they be matched?
How can the efficiency of the vision only SLAM algorithm be increased?

TABLE 3-2, RESEARCH QUESTIONS

Tests are performed on test data by means of an implementation written in Matlab[1], using the Vision only SLAM algorithm, described in the article by Sim et al. (2005), as a starting point. At first, a simple visual ego-motion estimator is implemented. To reduce the ego-motion estimation error, this implementation is replaced by a more advanced version available as a toolbox at TNO (Van der Mark, 2007), which is also used to calculate the 3D position of features. The SIFT feature extractor is written by Lowe (1999).

3.6 RELATED WORK

A lot of research has been done on SLAM by Montemerlo, Thrun, Durrant-Whyte, Bailey and many more (Montemerlo & Thrun, 2003) (Durrant-Whyte & Bailey, 2006). Most methods however depend on laser scanners to sense the environment and odometry to estimate the initial movement. Laser scanners provide very accurate and long-range measurements that can also be used for SLAM in outdoor environments. Other sensors that are typically used for SLAM are ultrasonic sensors. Ultrasonic measurements are short and coarse due to cone shaped beam patterns, which limits their use. Both sensors are active and therefore easy to detect. For a robot to remain undetected, passive sensors need to be used. A camera is such a sensor that is also used for SLAM. Additional to being passive it also has the advantage of having a high information content. By using algorithms, landmarks can be identified by their appearance.

The method described in this thesis is based solely on vision. This is done before, in an indoor setting (Sim, Elinas, Griffin, & Little, 2005). To make vision based SLAM more widely applicable as an enabling technique, more research is needed with fewer constraints. The experiments described in this thesis are performed in a large-scale outdoor environment. This has its effects on the selection, matching and storage of the landmarks, the degrees of freedom of the motion of the robot, the dynamics of the environment et cetera.

[1] Matlab®: http://www.mathworks.com/

3.7 Related methods

SLAM is not the only way to determine the position of the robot. In most cases, it is more accurate and economical to utilize the Global Positioning System (GPS). The accuracy of GPS is in the order of meters depending on the system used. By using a reference GPS with known coordinates, the positional uncertainty can be reduced to approximately 15 to 30 centimeter. This technique is called Differential Global Positioning System (DGPS). By incorporating real-time kinematics to DGPS, the measurement can even be improved to approximately 1 to 2 centimeter (Bailey, 2002).

An advantage of using GPS in comparison to SLAM is that the positional error is not cumulative. A disadvantage can be the dependence on the signal transmitted by satellites. The satellites need to be in the line of sight of the robot. Therefore, the signal is not available everywhere, for example behind large objects or indoors. In addition, the signal from the satellites can be actively jammed.

Another method to build map and determine the position of a robot in it, is to use pictures taken from above by an airplane or a satellite. This gives an overall picture of the environment. However, these pictures are not always available, complete or up to date. Furthermore, these images cannot be used indoor or in areas that are obscured, for example by trees.

3.8 Report overview

In part I of the thesis, chapters 4, 5 and 6 will first build up all necessary theoretical background, followed by the state of the art research in chapter 7. If the reader is already familiar with the general theory, only the last chapter is needed to understand the experiments. In part II, the contributions are outlined. In chapter 8, our approach for outdoor SLAM and the software implementation to be used for this experiment is explained. Chapter 9 and 10, present the experiments and the results. The conclusion and the discussion are presented in respectively chapter 11 and 12. Finally, suggestions for future research are given in chapter 13. This will focus on active vision and a 3D-reconstruction of the environment.

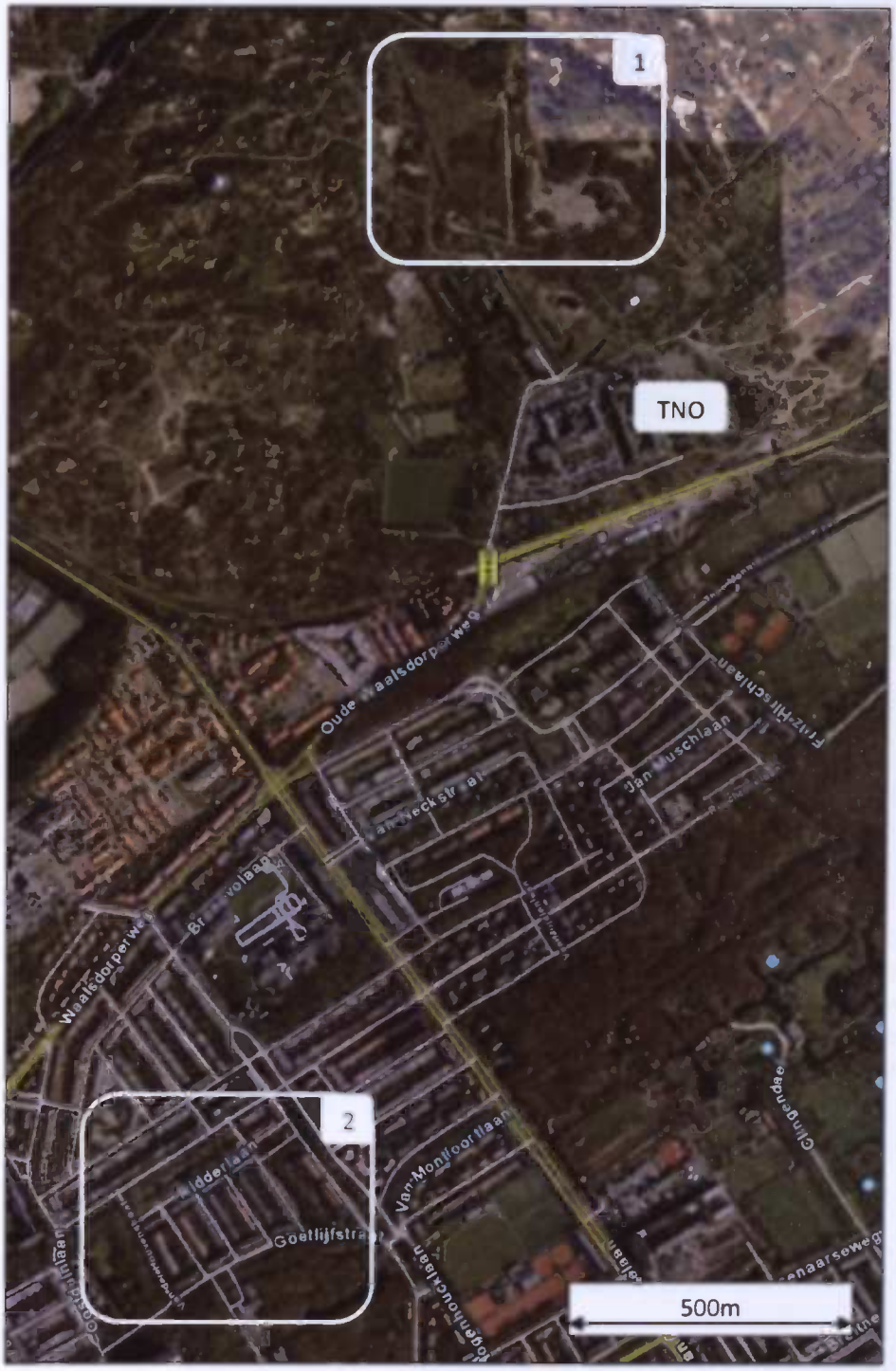

FIGURE 3-3, TEST AREAS NEAR TNO IN THE HAGUE, SECTION 1 = MILITARY TEST TERRAIN: THE WAALSDORPERVLAKTE, SECTION 2 = URBAN AREA. (FROM GOOGLE MAPS[2])

[2] Google earth: http://earth.google.nl/

PART I: THEORETICAL BACKGROUND

In this part of the thesis, theory concerning SLAM and related topics such as probability theory and machine vision will be given. Chapter 4 covers probabilistic filtering in general. Applying probabilistic filters to SLAM is the subject of chapter 5. In chapter 6, short introductions are given on robotic vision and robotic kinematics. The first part of this chapter is focused on depth estimation based on stereo vision and image feature descriptors. Chapter 7 goes into more detail on a SLAM method with a non-parametric filter. This method will be the starting point of the SLAM implementation used for the experiment, which discussed in the second part of the thesis.

4 PROBABILISTIC FILTERS

In theory, when the pose of the robot is known precisely at each moment in time, a map of the environment can be build relatively easy. If the robot can measure the position of the landmarks in its near surroundings, it can use this information to draw a local map. By continuing this along the route, the local maps can be combined to form a global map. In practice however, there is always an amount of uncertainty in the pose of the robot. Furthermore, there is uncertainty in the localization of the surrounding landmarks. The question is how to deal with these uncertainties in such a way that the error of the localization and mapping is minimized.

Because sensor readings are never perfect, this uncertainty has to be taken into account when the robot has to reason about its situation. In probabilistic filtering, uncertainty of a state is reflected by probability distributions. Therefore, calculations concerning these uncertainties can be solved using calculus of probability. The example below shows a global localization problem. It is analog to the known example of the robot in a hallway with three equal doors (Thrun, Burgard, & Fox, 2005).

4.1 GLOBAL LOCALIZATION EXAMPLE

In a global localization problem, a robot has a map of its environment and it has to localize itself on this map. Imagine a robotic vehicle traveling along a road. The vehicle has a map of this road with the surrounding trees on it. It is has no knowledge about its position and could therefore be everywhere on the road. The belief of the vehicle about its position on the map is indicated by the striped horizontal red line at the bottom of the figure, which has an equal overall height (see Figure 4-1). The total area underneath the line is equal to one.

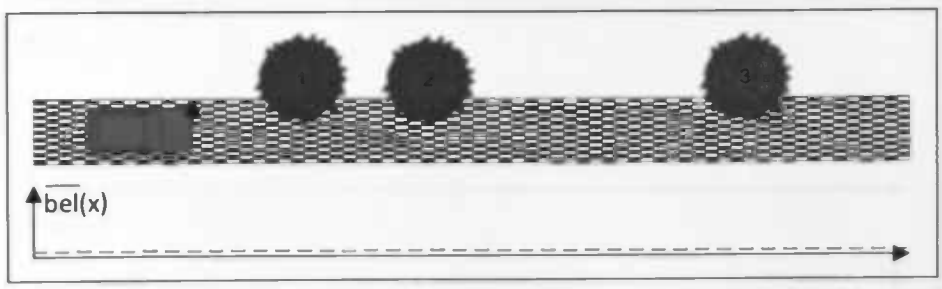

FIGURE 4-1, THE START POSITION OF THE VEHICLE. THERE IS NO KNOWLEDGE ABOUT THE POSITION OF THE VEHICLE, INDICATED BY THE HORIZONTAL LINE.

Suppose the vehicle has a camera that is pointed to the left, which gives it a limited local view. This is indicated by the small arrow on front of the car. By executing a control action, the vehicle moves several meters forwards and it observes a tree beside the road. The three trees on the map are indistinguishable so there is no way of knowing which tree it is. Anyhow, the probability of perceiving a tree is higher when being near a tree. This is indicated by the solid blue line (see Figure 4-2). The new belief can now be calculated from the old belief and the current probability. This is indicated by the striped red line.

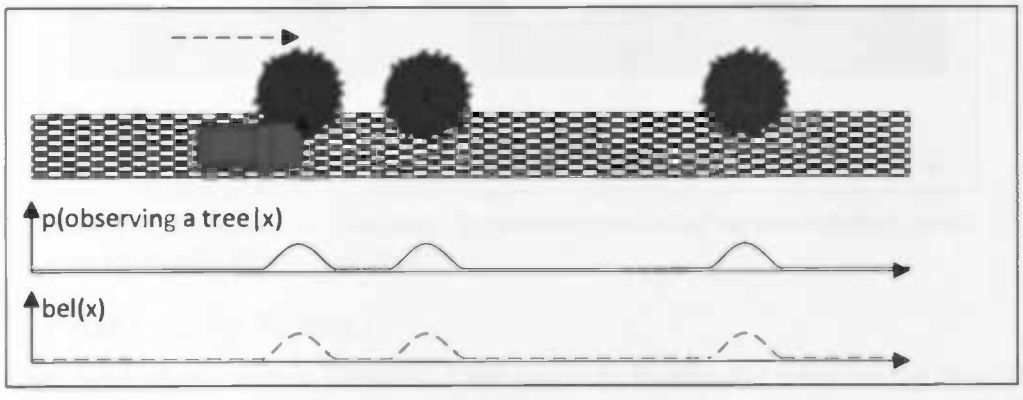

FIGURE 4-2, THE VEHICLE IS TRANSLATED AND A TREE IS OBSERVED NEARBY.

The vehicle executes a control action again, which is indicated by the striped arrow in Figure 4-3. While doing this, it moves its belief on the map by the same amount. In this way, it also has an initial belief of its position ($\overline{bel}(x)$) after the movement before doing a new observation. A control action introduces some uncertainty because the traveled distance is not precisely known and therefore the peaks of the striped-dotted red line are slightly lower.

It now observes a tree again so the probability of being near a tree is high again (see the solid blue line). The new belief about the robot's position $bel(x)$ can be deduced from the initial belief $\overline{bel}(x)$ and the current probability of being at a position given a tree has been observed $p(x|observed\ a\ tree)$. This is indicated by the lowest striped red line. The result is that the belief of being at the middle tree is highest.

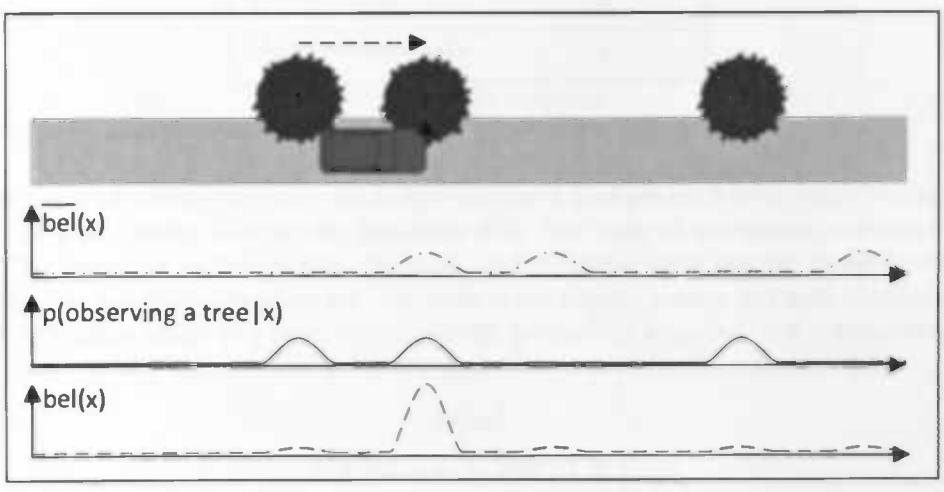

FIGURE 4-3, THE VEHICLE IS MOVED AND A TREE IS OBSERVED AGAIN.

When the vehicle travels further after executing a control action, the belief is also readjusted. In this manner, the vehicle still has knowledge about its position even when there is no tree to use as a beacon (see Figure 4-4). However, the uncertainty in it position becomes higher due to the control action.

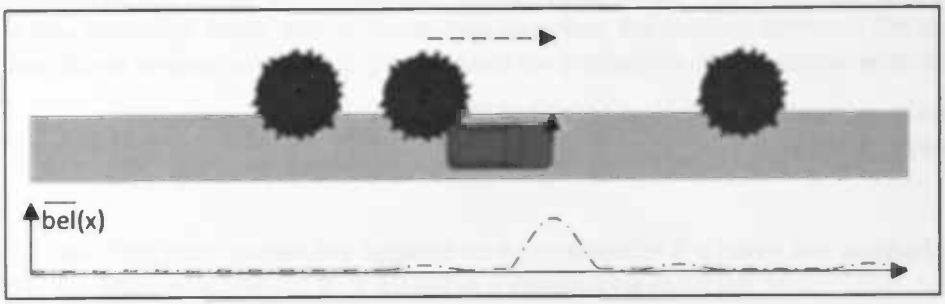

FIGURE 4-4, THE VEHICLE IS TRANSLATED. EVEN BEFORE OBSERVING THE ENVIRONMENT, IT HAS SOME KNOWLEDGE ABOUT ITS POSITION.

4.2 PROBABILITY THEORY

In the former example, the map is known and the task is to localize the robot on the map. In a case where SLAM is used, the map is not known yet. The localization and mapping are done simultaneously. The robot keeps track of its position and the map it has built up so far, which is called the state. In this context, the state includes everything about the robot and its environment that can influence the future. The state will be denoted x (see Table 4-1). If the state is the best predictor of the future, it is called complete. However, a state can be very complex, especially if the environment is dynamic. Therefore, only some variables of the state can be taken into account. This is called an incomplete state.

Data	Notation
State	x
Control data	u
Measurement data	z

TABLE 4-1, USED NOTATION

The state can be described by the begin value and the way it changes each time step. The begin situation is described by a probability distribution (equation 4-1). The state of the world is changed by control actions (u). The transition each time step, depends on the former state and the current control action. This is described by transition equation 4-2. The state is not known exactly, but with observations (z) the robot gets information about the state with a certain amount of accuracy. The measurements are assumed to be conditionally independent given the state. This is described by equation 4-3.

4-1) $$p(x_0)$$

4-2) $$p(x_t|x_{t-1},u_t) \quad for \quad t \geq 1$$

4-3) $$p(z_t|x_t) \quad for \quad t \geq 1$$

The goal is to calculate the state given the control actions and the measurements, which is called the posterior distribution. The posterior distribution can be calculated by a Bayes filter, which is a probabilistic technique that can be used to cope with uncertainties in observations. It is based on Bayes' theorem, which was invented by the British mathematician Thomas Bayes in the eighteenth century. Bayes' theorem, which is also known as Bayes' law or Bayes' rule describes the relation between the probability of an event A conditional on another event B ($p(A|B)$)and the probability of B conditional on A ($p(B|A)$).

4-4) $$p(A|B) = \frac{p(B|A)p(A)}{p(B)}$$ (Bayes' theorem, as mentioned in Ramoni & Sebastiani, 1999)

Where:

- $p(A)$ is called the prior probability because no information of B is taken into account
- $p(B)$ is the prior probability of B. It is used as a normalizing constant.
- $p(A|B)$ is the value to be calculated. It is called the posterior because it depends upon B.
- $p(B|A)$ is the conditional probability of B given A

In the case of SLAM, the information to be calculated (variable A in equation 4-4) is the position of the robot at each time step and the position of the landmarks. This is called the state and is assumed the total description of the situation. Variable B in equation 4-4 is the observation done at each time step. The observations depend only on the state and increase the knowledge about the state. On the other hand, control actions increase the uncertainty about the estimated movement of the robot and therefore decrease the knowledge about the state.

Example: Bayes' spam filter

Bayes' theorem is applied in many applications. In a spam filter, for example, it can be used to categorize incoming email as spam or not, by looking at the words it contains. The probability of an email being spam considering the words it contains is equal to the probability of finding those words in spam email, times the prior of receiving spam, divided by the prior probability of receiving an email with those words.

The word "Poker" for example is a common word in spam. When the dered email contains the word "Poker", the probability of the email being spam is higher. The word "the" is also a common word in spam but it is also common in non-spam email. The division by p(words) ensures a normalization. In general, when the prior probability p(spam) is higher, the chance of the current considered email being spam is also higher (see equation 4-5).

4-5) $$p(spam|words) = \frac{p(words|spam)p(spam)}{p(words)}$$ (spam filter)

Total probability

The total probability of an event A ($p(A)$) is the sum of the conditional probability of A given event B, times the probability of B, for all values of B (see equation 4-6).

4-6) $$p(A) = \int_{-\infty}^{\infty} p(A|B).p(B)\,dB$$ Continuous case

4-7) $$p(A) = \sum_{B} p(A|B).p\,(B)$$ Discrete case

The Markov Assumption

In a Bayes filter, variable x_t describes the total state at time t. If x_t is known then the past and future data are independent. This is because x_t is all there is to know about the present state and there cannot be more information from the past which could give more information about the future. This is called the Markov assumption, invented by the Russian Andrey A. Markov (1856-1922). This is a strong assumption. Things that are not included in x_t are assumed to have no influence.

4.3 Bayes' filter

In this section, the mathematical derivation of the Bayes' filter is given. Equations 4-8 and 4-9 follow from Bayes' theorem and the equation for total probability.

4-8) $$p(A|B) = \frac{p(B|A)p(A)}{\int_{-\infty}^{\infty} p(B|A').p(A')\,dA'}$$ Continuous case

4-9) $$p(A|B) = \frac{p(B|A)p(A)}{\sum_{A'} p(B|A').p(A')}$$ Discrete case

The denominator $p(B)$ of Bayes' rule (equation 4-4) is the same for every value of A' because it does not depend on A'. The part $(1/p(B))$ can therefore be written as a constant normalizer to simplify the Bayes' rule for calculations (see equation 4-10). After this calculation, the result must be normalized to 1.

4-10) $$p(A|B) = \eta.p(B|A).p(A)$$

The belief of the robot is the probability of the state at time t given all the measurements and all the control actions until time t (equation 4-11).

4-11) $$bel(x_t) = p(x_t|z_{1:t}, u_{1:t})$$ Belief

If the last measurement is not taken into account this is called the initial belief (see equation 4-12)

4-12) $$\overline{bel}(x_t) = p(x_t|z_{1:t-1}, u_{1:t})$$ Initial belief

By using the theorem of total probability (formula 4-6), where $A = x_t$ and $B = x_{t-1}$, conditioned on $z_{1:t-1}$ and $u_{1:t}$, the initial belief can be written as:

4-13) $$\overline{bel}(x_t) = \int p(x_t|x_{t-1}, z_{1:t-1}, u_{1:t}).p(x_{t-1}|z_{1:t-1}, u_{1:t})dx_{t-1}$$

The assumption is that the state is complete, so if x_{t-1} is known, the variables $z_{1:t-1}$ and $u_{1:t-1}$ give no extra information according to the Markov assumption. Therefore, $p(x_t|x_{t-1}, z_{1:t-1}, u_{1:t})$ can be written as $p(x_t|x_{t-1}, u_t)$. Furthermore, if the control action u_t is chosen randomly $p(x_{t-1}|z_{1:t-1}, u_{1:t})$ equals $p(x_{t-1}|z_{1:t-1}, u_{1:t-1})$. Now equation 4-13 can be written as:

4-14) $$\overline{bel}(x_t) = \int p(x_t|x_{t-1}, u_t).\, p(x_{t-1}|z_{1:t-1}, u_{1:t-1}) dx_{t-1}$$

This results in equation 4-15 that is called the prediction step. By calculating the initial belief from the former belief, the current control is taken into account.

4-15) $$\overline{bel}(x_t) = \int p(x_t|u_t, x_{t-1}).\, bel(x_{t-1})\, dx_{t-1}$$ Prediction step

The second part of the filter is used to calculate the new belief from the initial belief. The mathematical derivation of the second part starts with the definition of belief:

4-16) $$bel(x_t) = p(x_t|z_{1:t}, u_{t:1})$$ Belief

Using the Bayes formula and conditioning it on variables $z_{1:t-1}$ and $u_{1:t}$, results in equation 4-17.

4-17) $$p(x_t|z_{1:t-1}, u_{1:t}) = \frac{p(z_t|x_t, z_{1:t-1}, u_{1:t}).\, p(x_t|z_{1:t-1}, u_{1:t})}{p(z_t|z_{1:t-1}, u_{1:t})}$$

The denominator of the Bayes rule does not depend on x_t and can be written as a normalizer:

4-18) $$p(x_t|z_{1:t}, u_{t:1}) = \eta.\, p(z_t|x_t, z_{1:t-1}, u_{1:t}).\, p(x_t|z_{1:t-1}, u_{1:t})$$

The assumption is that the state is complete is exploited again:

4-19) $$p(x_t|z_{1:t}, u_{t:1}) = \eta.\, p(z_t|x_t).\, p(x_t|z_{1:t-1}, u_{1:t})$$

This results in equation 4-20, which is called the measurement update.

4-20) $$bel(x_t) = \eta.p(z_t|x_t).\, \overline{bel}(x_t)$$ Measurement update step

The measurement update step shows that by taking the last measurement that is taken into account, the new belief can be calculated from the initial belief.

In short, the Bayes filter update rule operates sequentially; the current state is calculated using the former state. This is done in two steps (see Table 4-2) as can be seen from the mathematical derivation of the Bayes filter that is shown above. First, the initial belief is calculated by using the control command (equation 4-15). After that, the new belief is calculated using the initial belief and the measurement (equation 4-20).

1	Bayes' filter $(bel(x_{t-1}), u_t, z_t)$
2	for all x_t do
3	$\overline{bel}(x_t) = \int p(x_t\|u_t, x_{t-1}).\, bel(x_{t-1}) dx_{t-1}$
4	$bel(x_t) = \eta.\, p(z_t\|x_t).\, \overline{bel}(x_t)$
5	endfor
6	return $bel(x_t)$

TABLE 4-2, THE BAYES FILTER ALGORITHM

In a Bayes filter, the state of a robot at time step t (x_t), depends on the former state (x_{t-1}) and the control command (u_t). The measurement (z_t) depends on the state (x_t). This is illustrated in Figure 4-5.

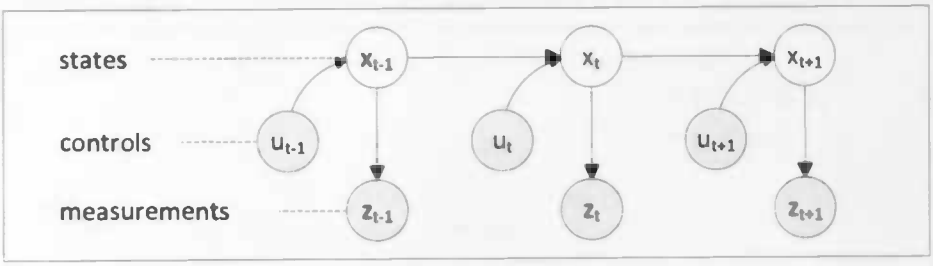

FIGURE 4-5, THE BAYES NETWORK: STATES, CONTROLS AND MEASUREMENTS (THRUN, BURGARD, & FOX, 2005)

4.4 PROBABILITY DISTRIBUTIONS

The values measured by the sensors of the robot in the global localization example (page 13), only approximate the real values. There is always an amount of noise in the measurements, which means the robot does not have an exact representation of the environment. This is also the case for the control action. There is uncertainty about the amount of movement a control action produced. The previous section is describes how these uncertainties can be dealt with in a concise way. However, in order to implement the Bayes filter, the probability distribution needs to be described by a function or a method. Parametric and non-parametric methods can be used. For both possibilities, the most common ones will be discussed in this subsection.

4.4.1 NORMAL DISTRIBUTION

The robot is uncertain about the information. However, there is knowledge about the amount of uncertainty. It is assumed that the uncertainty has a normal distribution, which is called Gaussian. In the scalar case, the one-dimensional normal distribution is calculated by equation 4-21.

4-21)
$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} . e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$
(THRUN, BURGARD, & FOX, 2005)

The normal distribution is defined by two variables; the mean and the variance. The mean (μ) is the mean value of the measurements if the measurement is repeated often. The variance (σ^2) resembles the degree of uncertainty of a measurement. Two distributions are shown as an example in Figure 4-6.

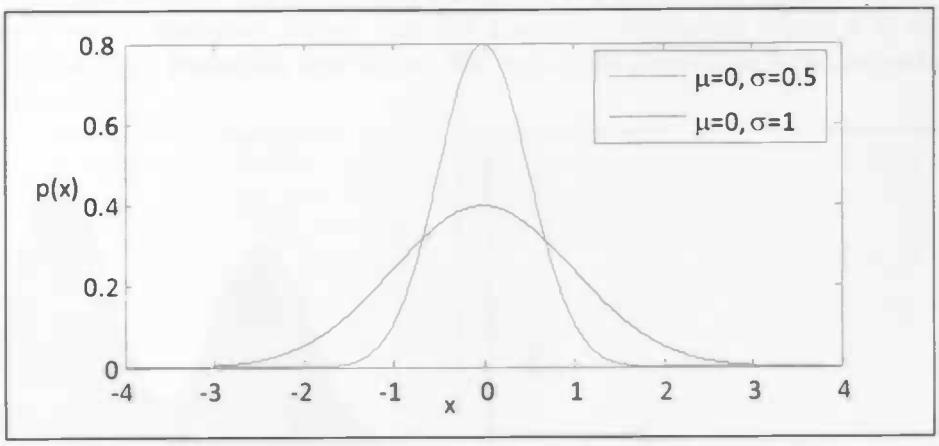

FIGURE 4-6, ONE-DIMENSIONAL NORMAL DISTRIBUTION

In the multidimensional case, the multivariate normal distribution is calculated by equation 4-22.

4-22)
$$p(x) = det(2\pi\Sigma)^{\left(\frac{-1}{2}\right)} . e^{\left(\frac{-1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)}$$
(Thrun, Burgard, & Fox, 2005)

The mean (μ) is a n-dimensional vector and the variance (Σ) is a $n * n$-dimensional covariance matrix. A two-dimensional example is given in Figure 4-7, where $\mu = \begin{bmatrix}0\\0\end{bmatrix}$ and $\Sigma = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}$.

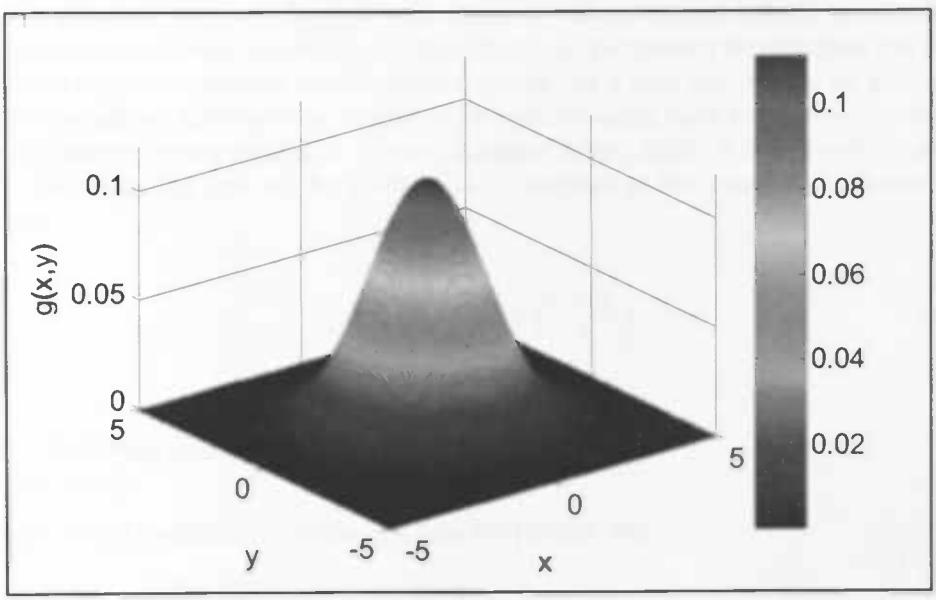

FIGURE 4-7, TWO DIMENSIONAL NORMAL DISTRIBUTION

4.4.2 NON-PARAMETRIC DISTRIBUTIONS

A Gaussian has a distribution that is fixed by two parameters. Non-parametric distributions also exist. Here the number and nature of the parameters are flexible and not fixed in advance, so the distribution can have any shape. A histogram (Figure 4-8) and a particle distribution (Figure 4-9) are both non-parametric estimates of a probability distribution. The probability distribution is illustrated by the curved red line in both figures.

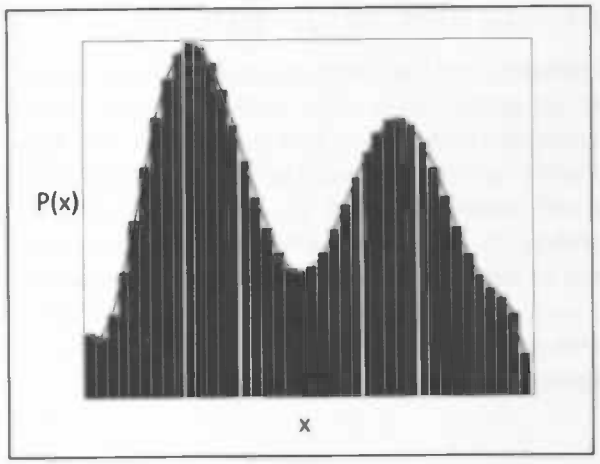

FIGURE 4-8, HISTOGRAM

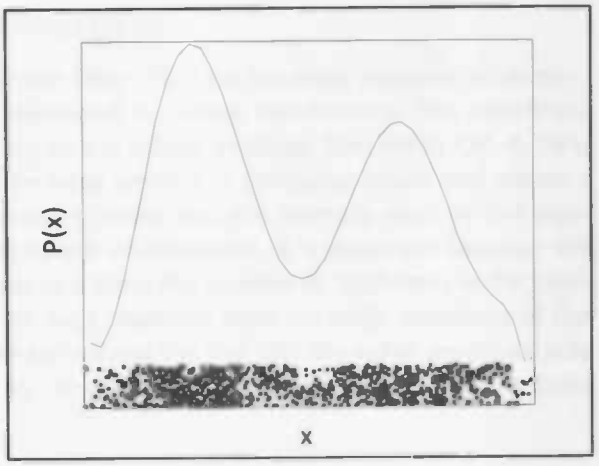

FIGURE 4-9, PARTICLE DISTRIBUTION

In the histogram, the height of the bar indicates the probability for that area. In a particle distribution, the horizontal axis is not divided into bins. The amount of particles in an area indicates the probability for that area. Because the distribution can have any shape, a particle distribution can be applied for all kinds of probability distributions that could not be estimated by a Gaussian distribution for example.

Histograms and particles represent discrete approximations of continuous beliefs. In some applications, however, a continuous estimate is needed. The density of non-parametric distributions can be estimated in several ways. A Gaussian estimate can be calculated, but this is only appropriate for a unimodal density. For multimodal sample distributions, a number of methods exist, such as the k-means algorithm, the density tree and kernel density estimation (Thrun, Burgard, & Fox, 2005). A kernel method estimates the probability at point x by the sum of the kernels, each centered at the n particle positions. It is formulated as follows:

4-23)
$$p(x) = \frac{1}{n * h} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$
(Webb, 2002)

Where:

- $K(z)$ = the kernel function
- h = the spread

Commonly used kernel functions for univariate data are (Webb, 2002):

- Rectangular: $\frac{1}{2}\ for\ |x| < 1, 0\ otherwise$
- Triangular: $1 - |x| for\ |x| < 1, 0\ otherwise$
- Normal: $\frac{1}{\sqrt{2\pi}} e^{\left(\frac{-x^2}{2}\right)}$

5 SLAM BASED ON A BAYES-FILTER

The assumption that the uncertainties can be expressed in terms of normal distributions is utilized by a number of implementations of the theoretical Bayes filter. As a result, several types of implementations of the Bayes filter exist to solve the SLAM problem. The major differences between these algorithms are the quality of the map and the amount of processing power and memory that is required.

5.1 KALMAN FILTER/EKF BASED APPROACHES

The Kalman Filter (Kalman, 1960) and the Extended Kalman Filter (EKF) are the most common implementation of the Bayes filter when uncertainties can be expressed in normal distributions. The algorithms result into the optimal map considering the inaccuracy of the sensor readings (Davidson, Cid, & Kita, 2004). A downside of the (Extended) Kalman Filter is the large amount of processor power and memory that is needed when using many landmarks. The processor power and the memory used by the algorithm scales quadratic with the amount of landmarks in the environment. It is quadratic because the landmarks are fully correlated. For EKF-SLAM to work in real time, the number of landmarks in the environment in which the robot has to navigate, have to be kept relatively small (< 100). Variations of the EKF can be used in environments with more landmarks by utilizing the fact that the robot perceives only part of the landmarks at each moment, for example by using sub-maps (Bailey, Nieto, Guivant, & Stev, 2006).

5.1.1 THE ALGORITHM

The Kalman filter and its extended version represent the belief about the state at time t by the mean (μ_t) and the covariance matrix (Σ_t) (see Figure 5-1).

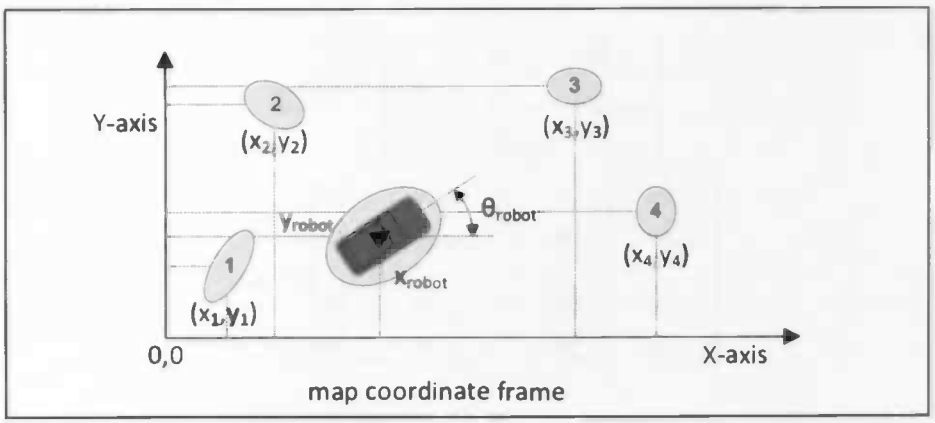

FIGURE 5-1, THE BELIEF ABOUT THE STATE IN A KALMAN FILTER IS COMPOSED OF THE MEAN VALUES AND THE COVARIANCE MATRIX OF THE POSITION OF THE ROBOT AND THE POSITION OF THE LANDMARKS WITH RESPECT TO THE MAP COORDINATE FRAME. THE UNCERTAINTY ABOUT THE POSITIONS OF THE ROBOT AND THE LANDMARKS IS ILLUSTRATED BY PROBABILITY ELLIPSES. BESIDE UNCERTAINTY ABOUT THE POSITION OF THE ROBOT, THERE IS ALSO UNCERTAINTY ABOUT THE ORIENTATION OF THE ROBOT.

The total mean (μ_t) consists of the mean of the robot pose (μ_{robot}) and all the means of the landmarks on the map (μ_{map}) (equation 5-1).

5-1) $$\mu_t = \begin{bmatrix} \mu_{robot} \\ \mu_{map} \end{bmatrix}$$

In the two dimensional case, where the robot moves on a flat surface, the robot mean (μ_{robot}) is defined by the mean x and y value and the mean robot orientation ($\hat{\varphi}_{robot}$) on the map (equation 5-2). The mean map matrix (μ_{map}) consists of all the mean x and y values of the n landmarks on the map (equation 5-3).

5-2) $$\mu_{robot} = [\hat{x}_{robot} \quad \hat{y}_{robot} \quad \hat{\varphi}_{robot}]^T$$

5-3) $$\mu_{map} = [\hat{x}_1 \quad \hat{y}_1 \quad \cdots \quad \hat{x}_n \quad \hat{y}_n]^T$$

The total covariance matrix (Σ_t) consists of the covariance matrix of the robot pose (Σ_{robot}) and the covariance matrix of the map (Σ_{map}) (equation 5-4).

5-4) $$\Sigma_t = \begin{bmatrix} \Sigma_{\text{robot}} & \Sigma_{\text{robot map}} \\ \Sigma^T_{\text{robot map}} & \Sigma_{\text{map}} \end{bmatrix}$$

The covariance matrix of the robot pose (Σ_{robot}) consists of a $3 * 3$ matrix, with the covariances of the x and y position and the orientation of the robot (equation 5-5). The covariance matrix of the map (Σ_{map}) consists of the covariances of all the x and y values of the landmarks (equation 5-6). If the number of landmarks is n, the dimension of Σ_{map} is $2n * 2n$

5-5) $$\Sigma_{\text{robot}} = \begin{bmatrix} \sigma^2_{x_r x_r} & \sigma^2_{x_r y_r} & \sigma^2_{x_r \varphi_r} \\ \sigma^2_{x_r y_r} & \sigma^2_{y_r y_r} & \sigma^2_{y_r y \varphi_r} \\ \sigma^2_{x_r \varphi_r} & \sigma^2_{y_r \varphi_r} & \sigma^2_{\varphi_r \varphi_r} \end{bmatrix}$$

5-6) $$\Sigma_{\text{map}} = \begin{bmatrix} \sigma^2_{x_1 x_1} & \sigma^2_{x_1 y_1} & \cdots & \sigma^2_{x_1 x_n} & \sigma^2_{x_1 y_n} \\ \sigma^2_{x_1 y_1} & \sigma^2_{y_1 y_1} & \cdots & \sigma^2_{y_1 x_n} & \sigma^2_{y_1 y_n} \\ \vdots & \vdots & \ddots & \vdots & \\ \sigma^2_{x_1 x_n} & \sigma^2_{y_1 x_n} & \cdots & \sigma^2_{x_n x_n} & \sigma^2_{x_n y_n} \\ \sigma^2_{x_1 y_n} & \sigma^2_{y_1 y_n} & \cdots & \sigma^2_{x_n y_n} & \sigma^2_{y_n y_n} \end{bmatrix}$$

The Kalman filter algorithm first calculates the initial belief, in what is called the prediction step, by using the mean and the covariance of the last time step and the motor control of the current time step (u_t) as input variables (equation 5-7). A mathematical derivation of the Kalman Filter is given in the book Probabilistic Robotics (Thrun, Burgard, & Fox, 2005).

5-7) $$p(\bar{x_t} \mid x_{t-1}, u_t)$$

This produces the initial mean ($\bar{\mu}_t$) and the initial covariance ($\bar{\Sigma}_t$) (see Table 5-1, line 2 and 3).

1	Kalman filter $(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$
2	$\bar{\mu}_t = A_t \mu_{t-1} + B\, u_t$
3	$\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$
4	$K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$
5	$\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$
6	$\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$
7	return μ_t, Σ_t

TABLE 5-1, THE KALMAN FILTER ALGORITHM

The initial mean and covariance are used in line 4 to calculate the Kalman gain (K_t). The Kalman gain specifies the weight with which the measurement at the current time step (z_t) is incorporated into the new belief (μ_t and Σ_t). This is done in the measurement update step (line 5 and 6) which uses the sensor model (equation 5-8) to weight the initial believe.

5-8) $$p(z_t | x_t, m)$$

The sensor model gives the probability of taking a measurement (z_t) given a pose (x_t) and a map (m). Because the probability of doing a measurement depends on the landmarks on the map, the landmarks can be included in the Bayes network diagram (Figure 5-2).

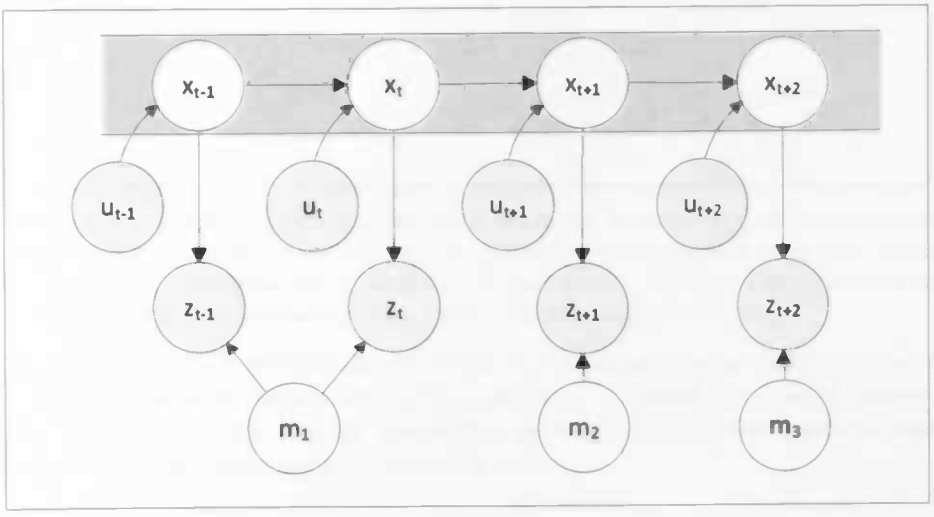

FIGURE 5-2, BAYES' NETWORK. THE STATE u_t DEPENDS ON THE FORMER STATE x_{t-1} AND THE ACTION COMMAND u_t. THE MEASUREMENT z_t DEPENDS ON THE STATE u_t AND LANDMARK THAT CAN BE PERCEIVED AT TIME t ON THE MAP m_1 (THRUN, BURGARD, & FOX, 2005)

5.1.2 DATA ASSOCIATION

In the example illustrated in Figure 5-2, the robot perceives landmark m_1 at pose x_{t-1} and x_t , landmark m_2 at pose x_{t+1} and landmark m_3 at pose x_{t+2} . The robot has to know, which landmark a measurement corresponds to. For example, that the landmark perceived in time step t is the same as the one perceived at time step $t-1$. This is called data association.

The Kalman filter depends heavily on a right data association (Montemerlo & Thrun, 2003). This is not trivial, especially in a natural environment. To make data association less difficult, the environment of the robot can be adapted by using artificial labels like bar codes for example, but this is not always desirable.

5.1.3 LINEARIZATION

The Kalman filter assumes the observations are linear functions of the current state and that the current state is a linear function of the previous state. It also assumes that the current state is represented by a Gaussian distribution.

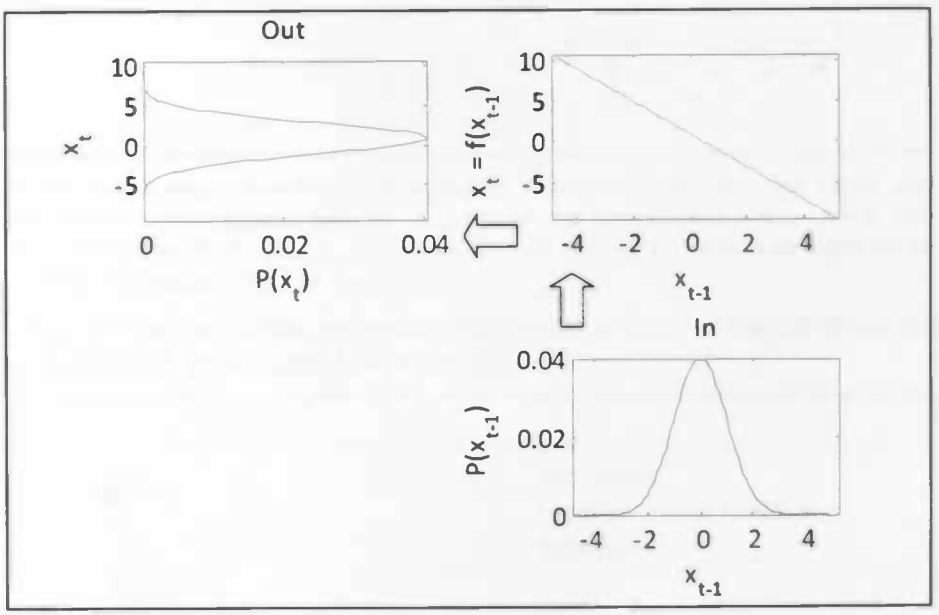

FIGURE 5-3 A LINEAR TRANSFORMATION OF A GAUSSIAN VARIABLE. THE BOTTOM GRAPH SHOWS A 1-DIMENSIONAL STATE AT TIME (x_{t-1}). THE TRANSFORMATION IS OF THE FORM ($x_t = a * x_{t-1} + b$) WHERE $a = -2$ AND $b = -1$ (TOP RIGHT). THE RESULTING STATE (x_t) IS A GAUSSIAN AGAIN BUT THE PARAMETERS, μ AND σ, HAVE CHANGED (TOP LEFT).

A Gaussian distribution results in another Gaussian distribution after a linear transformation (Figure 5-3). The parameters of this Gaussian distribution can be calculated in closed form, which makes the Kalman filter attractive. Unfortunately, the linearity assumption does not hold in most practical state transitions and measurement cases. Figure 5-4 shows an example of this.

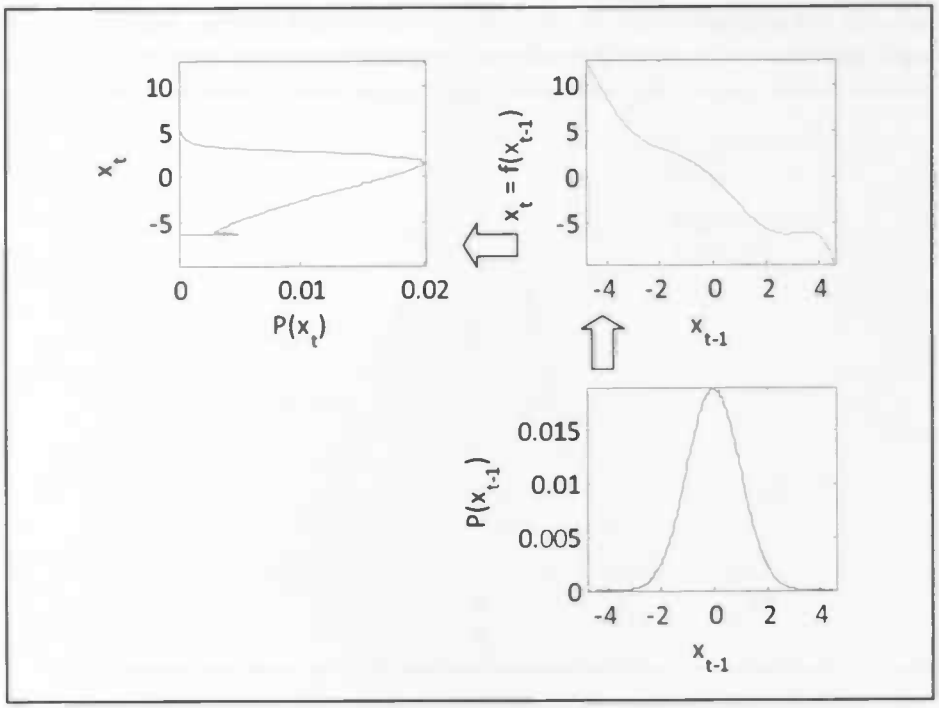

FIGURE 5-4, A NON-LINEAR TRANSFORMATION OF A GAUSSIAN VARIABLE. THE GREEN LINE (TOP RIGHT) IS NOT STRAIGHT BECAUSE IT IS NOT OF THE LINEAR FORM $y = a * x + b$. THE GAUSSIAN DISTRIBUTED POSE (x_{t-1}) IS TRANSFORMED INTO A NON-GAUSSIAN DISTRIBUTED POSE (x_t) AS SHOWN BY THE RED LINE (LEFT).

The Extended Kalman filter can handle this problem by the linearization of the non-linear transformation function (Durrant-Whyte & Bailey, 2006) (see Figure 5-5).

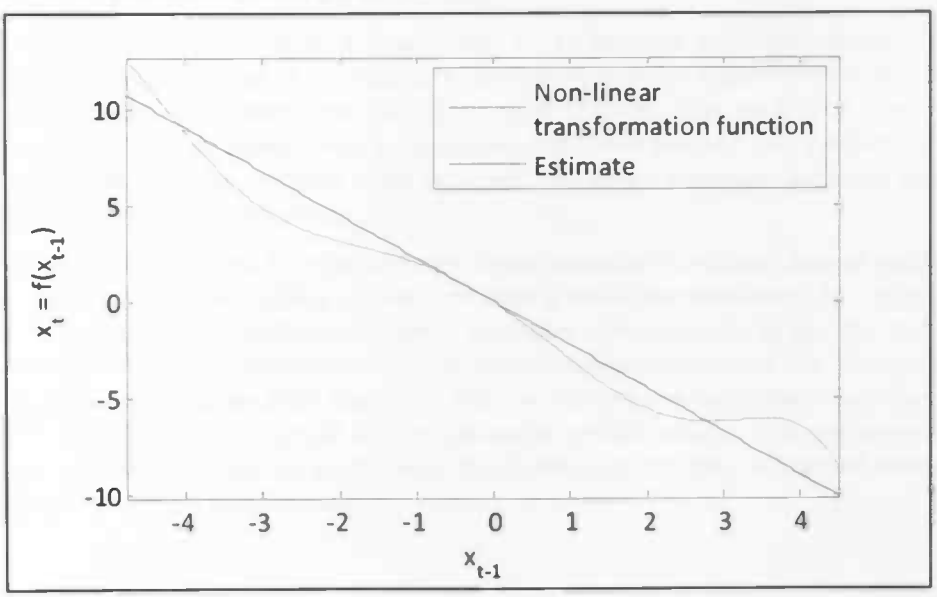

FIGURE 5-5, THE EXTENDED KALMAN FILTER CAN HANDLE THE NON-LINEAR TRANSFORMATION FUNCTION BY CALCULATING THE FIRST ORDER TAYLOR EXPANSION. THE SLOPE OF THE LINEAR APPROXIMATION IS CALCULATED AT THE MEAN VALUE OF THE INPUT DISTRIBUTION (x_{t-1})

The Extended Kalman Filter can also handle the non-linearity by retransforming the non-Gaussian distributed pose (x_t) into a Gaussian. One way to do this is to take the Monte-Carlo estimate (Figure 5-6).

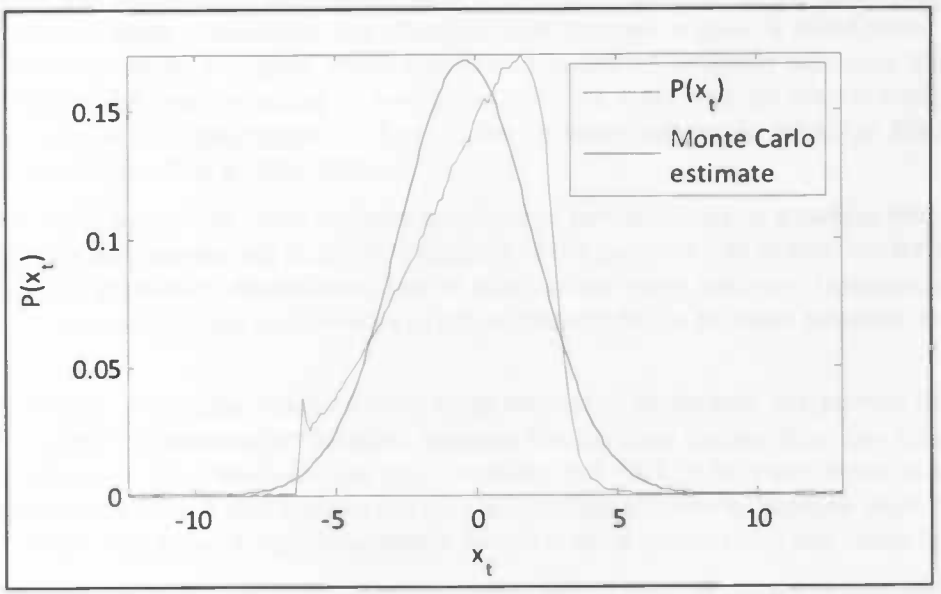

FIGURE 5-6, THE MONTE-CARLO ESTIMATE IS OBTAINED BY DRAWING MANY SAMPLES FROM THE NON-GAUSSIAN DISTRIBUTION ($P(x_t)$) AND CALCULATING THE MEAN AND COVARIANCE.

By making a non-linear transformation function linear, the EKF can be calculated in closed form. However, this linearization is also a disadvantage of EKF SLAM. The linearization estimate is in many cases not a good estimate, introducing an unacceptable large linearization error.

5.2 ADVANTAGES / DISADVANTAGES

A Kalman filter is an implementation of a Bayes filter. It can be used if the probability of the state is Gaussian distributed and the noise of the estimate movement and the measurements are independent and Gaussian distributed. In addition, the transitions from state to state each time step must be described by a linear function. By doing these assumptions, the SLAM problem has a closed form solution, which is an advantage. In most practical cases however, the linear transition between states is a too strong assumption, which is a disadvantage.

The Extended Kalman filter is used to cope with non-linear transition functions. Several methods exist to calculate the estimate of the probability distribution after a non-linear transition. An option is to calculate the linear estimate of the transition function at the point of interest and to use this to calculate the estimated Gaussian distributed probability after transition. Another option is to use multiple sampling to get the non-Gaussian distribution after transition and use statistics to calculate a Gaussian distributed estimate of this distribution. The Kalman and its extended version assume that the noise of the measurements have a Gaussian distributed probability. This is often not the case in practice and estimating it by a Gaussian distribution is a disadvantage.

5.3 SLAM USING A PARTICLE FILTER

As mentioned in the introduction, the Kalman filter is an optimal solution to the SLAM problem, given certain assumptions. These assumptions are often not valid. Another method is SLAM based on particle filters. Here the focus lies on the speed of the algorithm. Examples of solutions that solve the that solve the SLAM problem with many particles in (near) real time are FastSLAM 1.0 (Montemerlo, Thrun, & Koller, 2002), FastSLAM 2.0 (Montemerlo, Thrun, Roller, & Wegbreit, 2003), DP-SLAM (Eliazar & Parr, 2003) and σ-SLAM (Elinas, Sim, & Little, 2006).

These solutions solve part of the SLAM problem by means of particle filters. In a particle filter, the probability distributions are represented by the distributions of the particles (see subsection 4.4.2). By using more particles, the probability distributions can be approached more precisely. However, using more particles also increases processor and memory usage. A tradeoff has to be made between precision and speed.

Beside the possibility of working real time for a large amount of landmarks, the particle filter has the advantage of solving the linearization problem, because the particles configuration can adapt to every probability distribution. The transformation does therefore not have to be made linear as in EKF. This removes linearization errors. It also handles the data association problem by applying multi hypotheses, in a natural manner. This makes it a good candidate for SLAM using stereo vision and image features.

By representing the probability distribution by many particles, all kinds of shapes of distributions can be estimated. The solution to the SLAM problem needs to estimate the state that consists of pose of the robot and the position of the landmarks. In most cases, too many particles are needed to get a good estimate of both. A method is used to get a solution to the SLAM problem that is partly based on a particle filter and partly based on a Kalman filter. This will be discussed in more detail in the next section.

5.4 RAO-BLACKWELLIZATION

As a reminder: In SLAM, the complete state x_t is formed by the robot pose at time t (s_t) and the map at time t (m_t) (see equation 5-9). The goal is to estimate the state given the controls and the measurements (5-10). The state transition model and the measurement model are described by equation 5-11 and 5-12.

5-9) $$x_t = \{s_t, m_t\}$$

5-10) $$p(x_t | z_{1:t}, u_{0:t})$$

5-11) $$p(x_t | x_{t-1}, u_t)$$ State transition probability

5-12) $$p(z_t | x_t)$$ Measurement probability

Where:

- s_t is the robot pose at time t
- m_t is the map at time t
- z_t is the observation at time t and $z_{1:t}$ is the set of all observations until time t
- u_t is the control signal or a measurement of the motion of the robot from time $t-1$ to t and $u_{0:t}$ is the set of all controls until time t

As seen in section 5, the former equations can be transformed into equation 5-13 using Bayes' formula.

5-13) $$p(x_t|z_{1:t}, u_{0:t}) = \eta p(z_t|x_t) \int p(x_t|u_t, x_{t-1})\, p(x_{t-1}|z_{1:t-1}, u_{0:t-1}) dx_{t-1}$$ (Thrun, 2002)

Where:

- η is a normalizing constant

Inserting equation 5-14 into equation 5-13, results in equation 5-15.

5-14) $$Bel(x_t) = p(x_t|z_{1:t}, u_{0:t})$$ Believe of the robot

5-15) $$Bel(x_t) = \eta p(z_t|x_t) \int p(x_t|u_t, x_{t-1})\, Bel(x_{t-1}) dx_{t-1}$$

The believe of the robot consist of the pose of the robot and all perceived landmarks given all observations and all controls (equation 5-14). When using a particle filter, the probability distribution of the robot pose and the position of the landmarks are presented by particles. Therefore, the number of particles that are required can increase rapidly.

A solution to the problem of the need for too much particles for both the robot pose and the landmark positions, is factoring the SLAM posterior (Doucet, De Freitas, Murphy, & Russell, 2000). This is called Rao-Blackwellization (see equation 5-16). The posterior is factored, using a Rao Blackwellized particle filter, by sampling over all possible robot poses and marginalizing out the map (Murphy, 1999)(see equation 5-16). The main assumption here is that if the path is known exacty, then all landmark estimation problems are uncorrelated.

5-16) $$Bel(x_t) = p(s_t, m_t|z_{1:t}, u_{0:t}) = p(s_t|z_{1:t}, u_{0:t}) \prod_k \mathrm{p(m(k)}|s_{0:t}, z_{1:t}, u_{0:t})$$ (Sim, Elinas, Griffin, & Little, 2005)

Where:

- $m(k)$ is the k[th] landmark on the map

The stereo vision SLAM that is used in this experiment utilizes Rao-blackwellization. The localization sub problem is solved by using a particle filter (Elinas, Sim, & Little, 2006). The sub problem of the landmark estimation is solved by uncorrelated Kalman filters.

5.4.1 ADVANTAGES

Complexity

Because the positions of the landmarks are uncorrelated, the update complexity, each time step, is less complex compared to the Kalman filter, where all the landmarks are correlated. In the 2D case, the order is not quadratic $O(2n)^2$, but linear $O(n * 2)^2$.

Linearization

Kalman filters assume linearity in the motion and sensor model. The uncertainty in the motion model, which models the robot pose, is in many cases, in the long run, far from linear (Doucet, De Freitas, & Gordon, 2001). Rao-Blackwellized particle filters handle this nonlinearity by representing the pose of the robot by particles. The particles can easily be translated by directly using the transformation function because the distribution is non-parameterized.

Multi hypothesis data association

SLAM based on a Kalman filter or an extended Kalman filter depends on perfect identification of the perceived landmarks if no special adjustments to the algorithm are made. This is not the case for σSLAM, FastSLAM (Montemerlo, Thrun, & Koller, 2002) and its improved version, FastSLAM 2.0. These are all Rao-Blackwellized particle filters. In the particle filters, each particle has its own data association and therefore the algorithm can perform multiple hypotheses tracking (MHT). This ability to tackle the data association problem is a practical advantage of particle filters. Another practical advantage is the ease of implementation of a particle filter.

5.4.2 Disadvantages

Consistency

A great disadvantage of the σSLAM algorithm is its inconsistency in the long term (Bailey, Nieto, & Nebot, 2006). This means that the particle filter produces larger estimation errors than predicted by the model on which the filter is based. In their paper, Bailey et al. show that their Rao-Blackwellized particle filter algorithm also degenerates over time (Bailey, Nieto, & Nebot, 2006).

Coalescence

The degeneration phenomenon is a common problem in particle filters, where after a few iterations; all but one particle will have negligible weight (Arulampalam, Maskell, Gordon, & Clapp, 2002). When using re-sampling, this problem is also reverted to as coalescence where, in the long run, all the particles have a common ancestor particle. Degeneration takes place regardless of the amount of landmarks in the environment or the number of particles used. Bailey, Nieto and Nebot (2006) found that Rao-Blackwellized particle filters, in the long term, always produce an optimistic estimate of uncertainty.

6 ROBOTIC VISION AND KINEMATICS

The former section covered the theory on how to estimate the state using observations and control actions. This section discusses how this information can be obtained.

6.1 OBSERVATIONS

For a robot to be useful, it has to be able to observe its environment. To navigate for example, a robot has to make a distinction between obstacles and drivable terrain. In many cases, the position of the obstacles is not given beforehand. In that case, the only way for the robot to get this information is by measuring its surroundings by means of its sensors.

When using vision, image features can be used to detect landmarks. A general method for finding and describing image features will be discussed first. Later on, a more specific description of a scale invariant image feature transformation that is used in this experiment will be given. When the image features are found, the location of the landmark to which the feature corresponds needs to be estimated. This will be the subject of subsection 6.4.

6.2 IMAGE FEATURES

Finding corresponding image features of two images of the same scene is an important subject in the area of computer vision. It can be divided in three sub tasks which are described below; detection, description and matching.

Detection

First, an interesting point has to be localized/detected. The repeatability of this detection is important. The detector has to be able to detect the same point during different conditions such as changing lighting conditions, different viewing angles and different distances et cetera. Many different detectors have been proposed. Detectors can detect the "interesting points" of an image by searching for corners, blobs or t-junctions et cetera. Well know detectors are the Harris corner detector (Jung & Rad, 2001) and the Hessian detector (Moreels & Perona, 2005).

Description

After the detector has located an interesting point, its surrounding area can be described to distinguish it from the other points. The description has to be distinctive and it has to be robust enough to not be too much effected by certain changes such as noise, affine transformations or changing lighting conditions. The amount of required invariance to these changes has great influence on the choice of which descriptor to choose for a certain application. Many different techniques have been presented in the literature. Mikolajczyk en Schmid did a survey of local descriptors in the context of matching and recognition (Mikolajczyk & Schmid, 2005).

Matching

When the descriptors of the features in two pictures are known, the similarities of the picture can be expressed by the distance between these descriptors. Not all earlier perceived features on the map have to be compared to the current perceived features, only those that are close according to the map.

6.3 SCALE INVARIANT FEATURE TRANSFORMATION

Object recognition is the method by which is determined if a current observed object has been observed before. Modeling and recognizing landmarks can help to close loops in outdoor SLAM (Ramos, Nieto, & Durrant-Whyte, 2007). A powerful method to find the correspondence between perceived landmarks will be calculated by means of image features that are scale invariant. These features are calculated by scale invariant feature transformations, which is abbreviated by SIFT (Lowe, 1999) (Lowe, 2004) (Se, Lowe, & Little, 2002). SIFT can be very distinctive but it is relatively slow to calculate (Bay, Tuytelaars, & Van Gaal, 2006). Another one is SURF, which is the abbreviation of: Speeded up robust features (Bay, Tuytelaars, & Van Gaal, 2006). If the cameras do not rotate too much about the z-axis (roll movement) during the run, an "upright" version like U-SURF could be sufficient also.

6.3.1 DETECTION

This section covers the SIFT implementation of Lowe (Lowe, 1999). SIFT uses a 2D Gaussian filter (with $\sigma = \sqrt{2}$) to find the keypoint locations (see formula 6-1). The continuous version can be seen in Figure 6-1.

6-1) $$G(x, y) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-x^2+y^2}{2\sigma^2}}$$

Because the image is discrete (not continuous) the filter also needs to be discrete. A kernel with 7x7 sample points will be sufficiently accurate (see Figure 6-2).

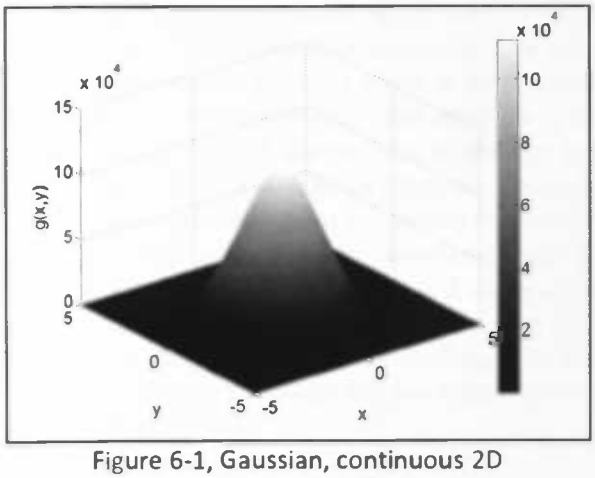

Figure 6-1, Gaussian, continuous 2D (with $\sigma = \sqrt{2}$)

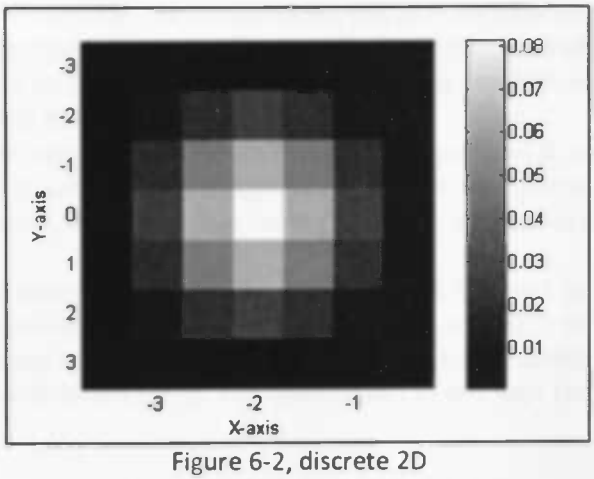

Figure 6-2, discrete 2D (with $\sigma = \sqrt{2}$ and size is $7 * 7$)

The SIFT algorithm finds the key locations by processing each image in a few steps at different scales (see at Figure 6-4 page 34). The steps per scale are discussed here, starting with image A, which is the input image.

- First, the input image (A) is smoothed by taking the convolution of the input image and the filter kernel. After that, the smoothed image (B) is subtracted from image A. The result is a so-called "difference of Gaussian" (C) (see also Figure 6-3).

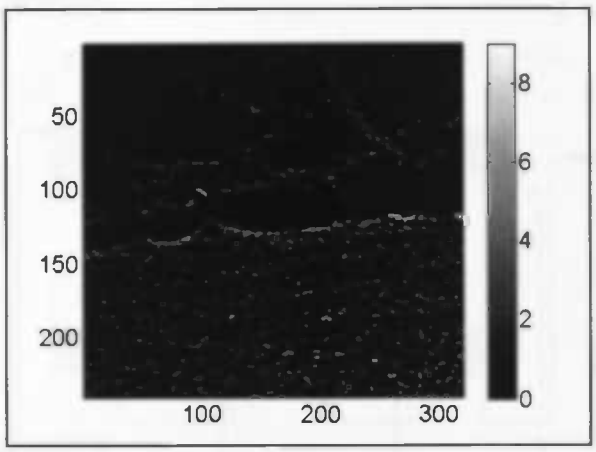

FIGURE 6-3, DIFFERENCE-OF-GAUSSIAN. NOTICE THE GRAY LEVEL RANGE, WHICH IS ADJUSTED FROM 0-255 TO 0-9.

- The difference-of-Gaussian is used as input to compute, the local maxima (D) and the local minima (F). These are calculated with a neighborhood of eight, which are the horizontal, vertical and diagonal adjacent pixels. A pixel is called a local maximum or minimum if it has the highest respectively the lowest value when compared to its eight neighboring pixels.
- Images D and F are rescaled to the size of the original image, which results in image E and G. In image D, E, F and G, all pixels are set to one to when they are regional maxima/minima and to zero otherwise. After processing the current level, the input image for the next level is calculated by re-sampling from the smoothed image B.
- The image is re-sampled from B using bilinear interpolation with a pixel spacing of 1.5 in each direction. This input image, called A' can be smoothed again to get B' et cetera. The process is repeated until the width or the height of the input image is equal to one. After the local maxima (E) and local minima (G) are calculated at the different scales, the localizations of the keys can be extracted.

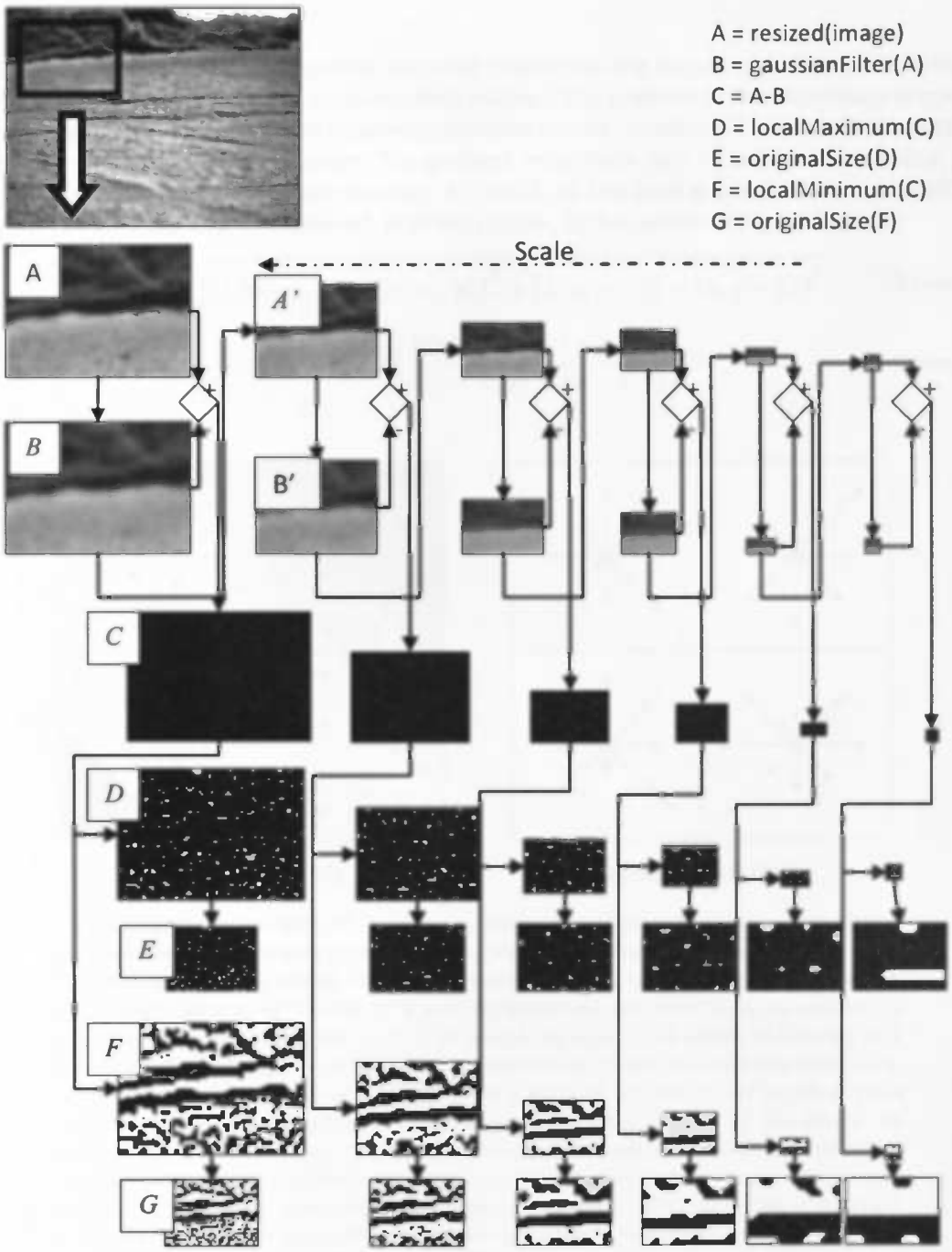

FIGURE 6-4, LOCAL EXTREMA LOCALIZATION

6.3.2 DESCRIPTION

The local image properties of the keypoint, are used to describe the keypoint at the scale it is found, so all computations are performed in a scale-invariant manner. The gradient of the local image properties is calculated to get the orientation. The keypoint descriptor can be represented relative to this rotation to achieve an invariance to image rotation. The gradient magnitude $m(x, y)$ and orientation $\theta(x, y)$, are calculated by the pixel differences (see equation 6-2 and 6-3). The local gradients of the pixel difference result into the SIFT descriptor as explained, in broad outline, in the caption of Figure 6-5.

6-2) $$m(x,y) = \sqrt{\big(L(x+1,y) - (x-1,y)\big)^2 + \big(L(x,y+1) - (x,y-1)\big)^2}$$ (Lowe, 2004)

6-3) $$\theta(x,y) = \tan^{-1}\left(\frac{L(x,y+1) - (x,y-1)}{L(x+1,y) - (x-1,y)}\right)$$ (Lowe, 2004)

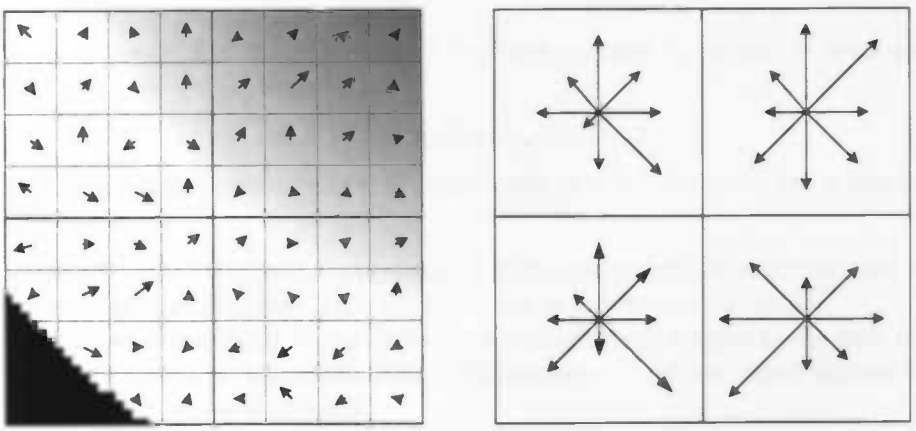

Image gradients Keypoint descriptor

FIGURE 6-5 SHOWS, ON THE LEFT, THE IMAGE GRADIENTS CALCULATED AT EACH IMAGE SAMPLE POINT AROUND THE KEYPOINT LOCATION. THESE ARE WEIGHTED BY A GAUSSIAN, INDICATED BY THE GRAY AREA THAT IS FADING TO THE PERIPHERY. THE RESULTING ARROWS PER 4X4 AREA'S ARE SUMMED INTO A HISTOGRAM OF 8 BINS REPRESENTING THE DIRECTIONS, AS SHOWN ON THE RIGHT. TO AVOID BOUNDARY EFFECTS, IN WHICH THE DESCRIPTOR ABRUPTLY CHANGES AS A SAMPLE SHIFTS SMOOTHLY FROM BEING IN ONE HISTOGRAM TO ANOTHER OR FROM ONE ORIENTATION TO ANOTHER, TRILINEAIR INTERPOLATION IS USED TO DISTRIBUTE THE VALUE OF EACH GRADIENT SAMPLE INTO ADJACENT HISTOGRAM BINS. THE TOTAL LENGTH OF THE ARROWS IN EACH OF THE 2X2 AREA OF THE KEYPOINT DESCRIPTOR IS NORMALIZED TO UNIT LENGTH TO REDUCE THE EFFECTS OF ILLUMINATION CHANGES. THE KEYPOINT DESCRIPTOR IN THE IMAGE SHOWS 2X2X8 ARROWS. EXPERIMENTS PERFORMED BY LOWE (2004) SHOW THE BEST RESULT IS ACHIEVED WITH A 4X4 ARRAY OF HISTOGRAMS OF 8 ORIENTATION BINS IN EACH. THEREFORE, THE EXPERIMENTS IN THIS THESIS USE A 4X4X8 = 128 ELEMENT FEATURE VECTOR FOR EACH KEYPOINT. (LOWE, 2004)

6.3.3 MATCHING

The descriptor of a SIFT feature consist of a 128 dimensional vector, describing the pixel orientation in relation to the main orientation. It is possible to calculate the amount of difference between two SIFT descriptors. Some methods for calculating vector distances are:

The city block distance:

6-4) $$dist = \sum_{i=1}^{128} (descr_{a,i} - descr_{b,i})$$

The euclidian distance:

6-5) $$dist = \sqrt{\sum_{i=1}^{128} (descr_{a,i} - descr_{b,i})}$$

The ratio between the angles of the inproduct of the entity vectors is, in case of small angles, a good approximation for the ratio of Euclidian distances:

6-6) $$dist = arccos(descr_a \cdot descr_b)$$

The descriptor distance values can be used in multiple ways to match features. The methods that will be used in the experiment are:

- An absolute distance threshold: If the distance of the perceived feature to the best matching feature on the map is smaller than a certain threshold, it is considered a match.
- A relative distance threshold: If the distance of the perceived feature to the best feature on the map is smaller than a certain percentage of the distance of the perceived feature to the second best match, it is considered a match.

Matching examples

Figure 6-6 illustrates the matching of SIFT features of two images taken at different time steps.

FIGURE 6-6, SIFT MATCHES FOR IMAGES OF THE DATA SET TAKEN AT THE WAALSDORPERVLAKTE, AT TWO DIFFERENT TIME STEPS. FRAME 1 AND FRAME 1100 ARE NOT TAKEN AT EXACTLY THE SAME POSITION AND THEREFORE THE FOREGROUND DIFFERS. THE BACKGROUND HOWEVER IS THE SAME. ALTHOUGH THE VIEWING ANGLE DIFFERS SLIGHTLY, THIS IS INDEED WHERE THE FEATURES MATCH, AS INDICATED BY THE LINES.

6.4 FROM IMAGE FEATURES TO 3D LANDMARKS

A stereo camera will be used to sense the environment. The main advantage of using stereo vision as opposed to mono vision is that distance information can be derived by stereopsis (Scharstein, Szeliski, & Zabih, 2002). This means that the distance to an object can be calculated by looking at the disparity between its position in the pair of left and right images. Other depth cues, which are available when using either mono or stereo vision, are for example: changes over time like motion parallax, texture gradient and differences in lighting.

Apart from using a stereo or a normal camera, a camera with a wide-angle lens can be used to have a wider field of view. The wider field of view increases the angle in which objects can be perceived and therefore increases performance of the SLAM algorithm (Davidson, Cid, & Kita, 2004). However, the precision in depth estimation is reduced. By using an omni-vision camera, the field of view is even larger (Kwok & Rad, 2006) (Bailey, 2003). In that case, the entire area around the robot (360°) can be viewed.

Before using the stereo camera as a distance measurement tool, the following steps are performed. First, the camera is calibrated and the intrinsic and extrinsic parameters are obtained. Then, by using the intrinsic parameters, the images can be rectified. After this, by the extrinsic parameters, the disparity and depth is calculated. These steps are outlined below.

6.4.1 THE PINHOLE CAMERA

The pinhole model can be used to calculate the projection of 3D scene space onto a 2D image plane (see Figure 6-7). The pinhole model is a simplified model in which the camera center is an infinitely small hole through which the light beams pass. The image plane is placed in front on the camera.

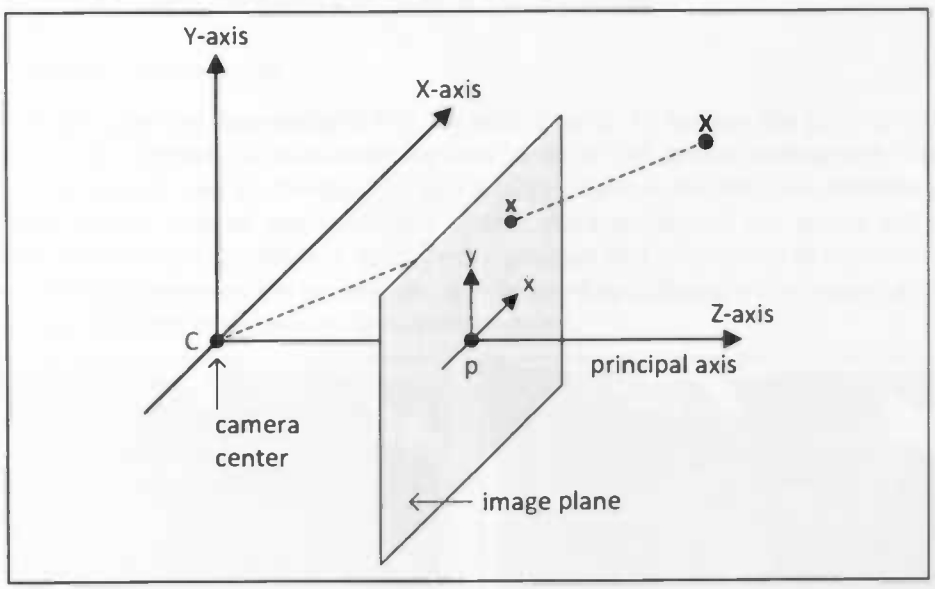

FIGURE 6-7, THE PINHOLE CAMERA GEOMETRY, C IS THE CAMERA CENTER AND P THE PRINCIPAL POINT (HARTLEY & ZISSERMAN, 2000).

$\boldsymbol{X} = [X, Y, Z]^T$ is assumed to be a point in space. When drawing a line from the camera center (the origin) to point $\boldsymbol{X}$, the line will cross the image plane at the position of image point $\boldsymbol{x}$. If the distance between the origin and the image plane is f, the image point is: $[(f.X/Z), (f.Y/Z), f]^T$. The Z-coordinate can be ignored, so the central projection mapping from world to image coordinates is:

6-7

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \xrightarrow{projection} \begin{bmatrix} \left(\frac{f.X}{Z}\right) \\ \left(\frac{f.Y}{Z}\right) \end{bmatrix}$$

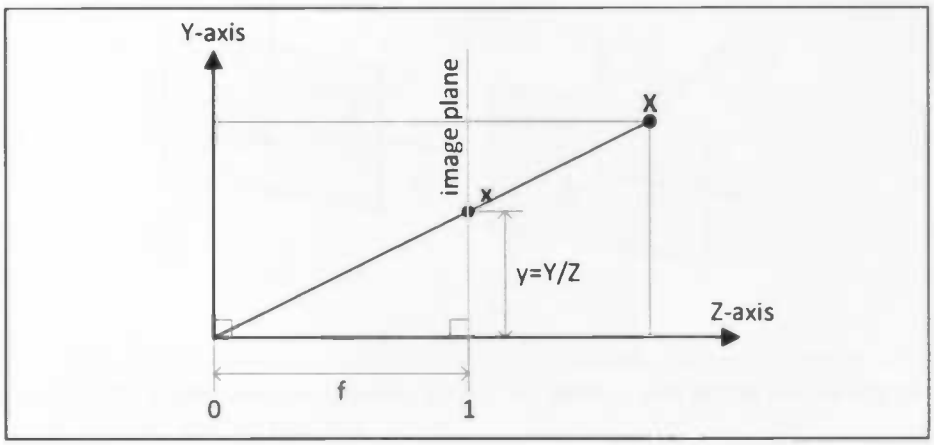

FIGURE 6-8, IF THE PROJECTION CENTER IS SITUATED IN THE ORIGIN OF THE WORLD FRAME AND THE IMAGE PLANE IS $Z = 1$ THEN $\boldsymbol{x} = [x, y] = [(X/Z), (Y/Z)]$.

6.4.2 CAMERA CALIBRATION

The camera can perceive the environment and the objects in it. To deduce the position of the objects from the images, it is necessary to know what the parameters of the camera settings are. These parameters, as mentioned earlier, can be divided into two groups: internal and external. Internal (or intrinsic) parameters describe the internal geometric and optical characteristics of the lenses and the imaging device. External (or extrinsic) parameters describe the position and orientation of the camera's relative to each other. In this experiment, the parameters are obtained by calibrating the stereo camera with the calibration toolbox that was developed by Jean-YvesBouguet[3].

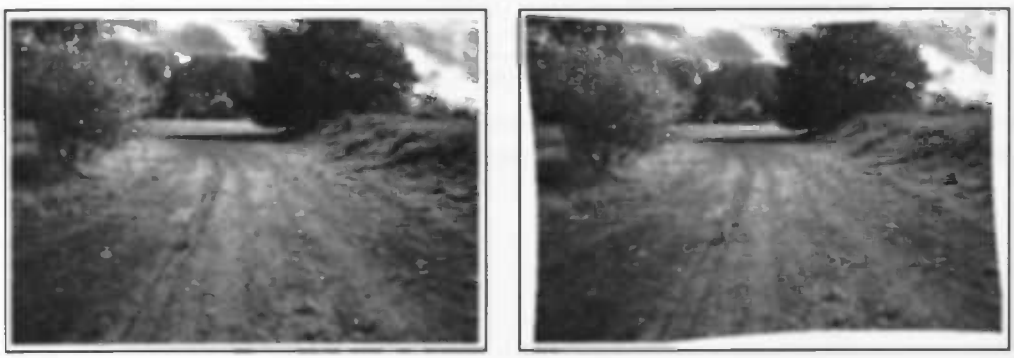

FIGURE 6-9, RECTIFICATION. WHEN THE CAMERA INTERNAL PARAMETERS ARE OBTAINED, THE IMAGES CAN BE RECTIFIED. THE IMAGE ON THE LEFT IS THE ORIGINAL, THE RIGHT IS RECTIFIED

[3] Camera calibration toolbox: http://www.vision.caltech.edu/bouguetj/calib_doc

6.4.3 DISPARITY/ DEPTH CALCULATION

The depth of a point can be calculated from a stereo pair of images. This is illustrated by Figure 6-10.

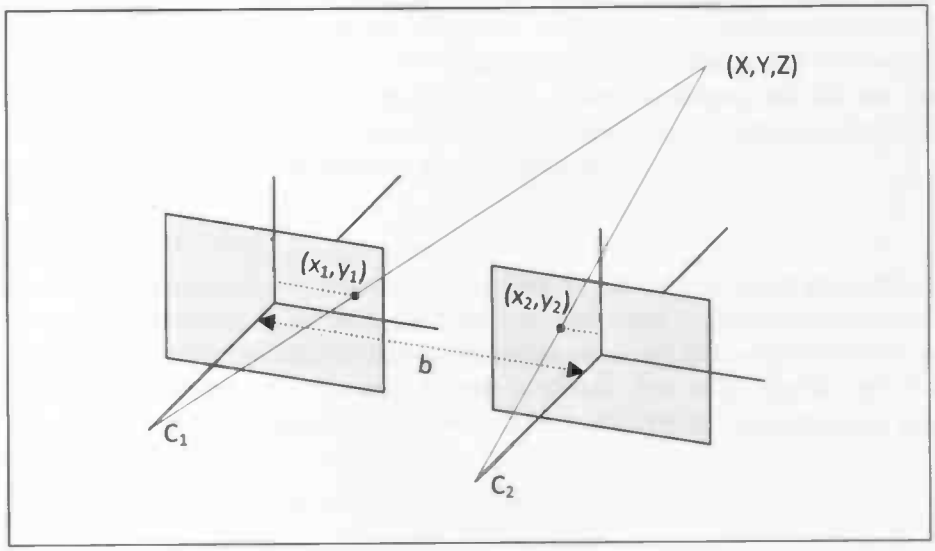

FIGURE 6-10, STEREO IMAGING (DAVIES, 2005). THE OPTICAL AXES OF THE SYSTEM ARE PARALLEL. VARIABLE b IS THE BASELINE.

The difference of the location of an object in the left and the right image is called the disparity (equation 6-10).

6-8)
$$x_1 = \frac{\left(X + \frac{b}{2}\right).f}{Z}$$

6-9)
$$x_2 = \frac{\left(X - \frac{b}{2}\right).f}{Z}$$

6-10)
$$D = x_1 - x_2 = \frac{b.f}{Z}$$
Disparity

From the disparity, the distance to the object (Z) can be calculated

6-11)
$$Z = \frac{b.f}{(x_1 - x_2)}$$
(Davies, 2005)

The actual method used, which applies optimal correction, is more complex. This is outside the scope of this thesis. For more information: "Statistical optimization for geometric computation" (Kanatani, 2005).

6.4.4 CORRESPONDENCE PROBLEM

To calculate the disparity between points from the left and right image, the algorithm has to determine which left image features correspond to the right image features. There are several ways to solve the correspondence problem. The SIFT features, discussed in the previous subsection, could be used to tackle it for example. For depth calculation, the image pairs per time stamp will be compared. Because changes in the images occur gradually, information from subsequent images can also be used. SLAM is a method to incorporate this information. The correspondence problem is explained earlier in the section about SLAM (section 5.1.2) where it is revert to as data association.

6.4.5 UNCERTAINTY MODEL

The certainty of measurements is not constant. It is proportional to a constant divided by the distance from the robot to the landmark. Therefore, the measurement noise increases with the distance of the camera to the landmark. It has an elliptic shape. The angle, at which the landmark is perceived, can be measured more accurately then the distance to the landmark. The noise spread will therefore have shapes like the shaded ellipses in Figure 6-11. This is called anisotropic homogeneous (Van der Mark, 2007, p. 23) (Kanatani, 2005).

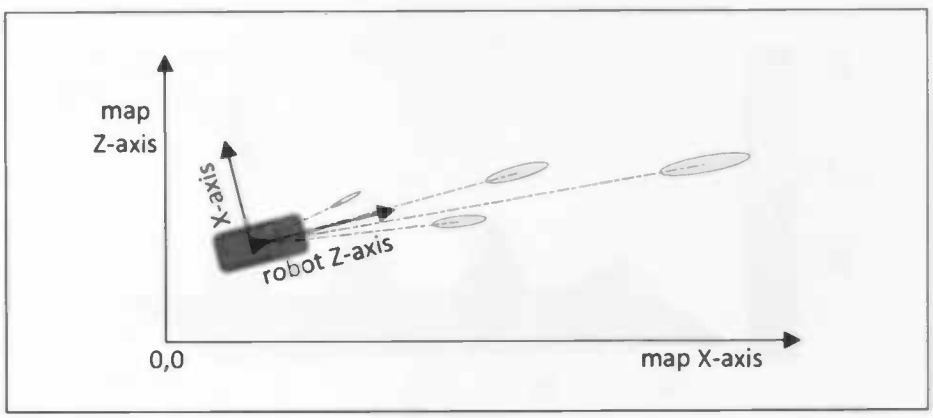

FIGURE 6-11, THE UNCERTAINTY IN THE MEASUREMENT INCREASES WITH THE DISTANCE TO THE LANDMARK. THE ANGLE AT WHICH THE LANDMARK IS PERCEIVED IS DETERMINED MORE ACCURATELY THAN THE DISTANCE TO THE LANDMARK. ALTHOUGH DISPLAYED IN TWO DIMENSIONS, THE ERROR IS ACTUALLY THREE DIMENSIONAL.

6.5 Robotic kinematics

Additional to perceiving, a robot must be able to manipulate its environment. By using its actuators however, the robot introduces uncertainty because of noise and external influences that are not modeled. A robot driving on sand, for example, does not precisely know its movement after a motor command has been executed, even if it exactly knows the amount of wheel rotations from its odometry sensors. The reason for this can be the slipping of its wheels, the timing between the rotations of the different wheels, the fluctuation in tire pressure and numerous other things.

6.5.1 Odometry

When the robot knows its initial position, it will lose positional information while if it moves without being able to measure its movement precisely. The uncertainty of the position of a robot is illustrated by Figure 6-12. Suppose the environment of the robot is a planar surface. The position of the robot is described by a x and y value. In this example, the mean start position $[x, y]$ is $[0, 0]$ and the uncertainty is small as illustrated by the steep left Gaussian curve. The robot now executes the command of moving 10 meters along the x-axis. Because of noise, the traveled distance is not precisely known. It is more or less 10 meter in the x-direction and additionally there can be a deviation on the y-axis.

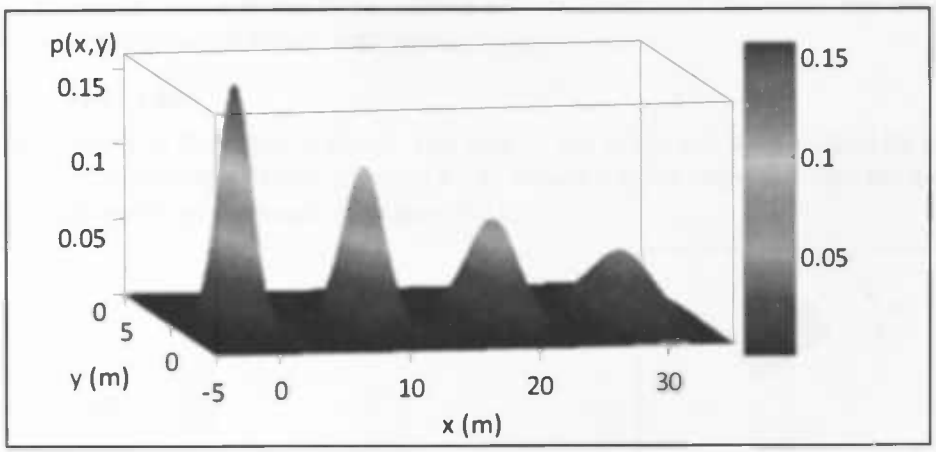

Figure 6-12 shows the positions of the robot by summing up the probability density functions of its four positions in time. The robot starts at position [0,0] and the uncertainty is small. Then, it moves about 10 meters to the right three times. The movement introduces additional uncertainty. This can be seen by the distribution, which is spread out more and more.

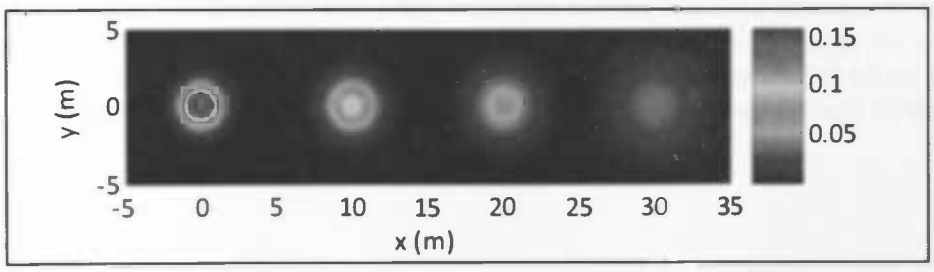

Figure 6-13, position of the robot, top view

The new position has a mean of $[10,0]$ but the uncertainty has grown. This is illustrated by the second Gaussian curve from the left. As can be seen, the second curve is lower than the first. The curve is also wider because the total probability equals one (see Figure 6-13). A lower peak means that the probability of the robot being at position $[10,0]$ is smaller. The robot can execute the same movement command again, which results in the third Gaussian curve. The next execution gives the fourth curve et cetera. The top of the curve gets lower and wider, which shows that the position of the robot is less well known. The uncertainty grows cumulatively.

One-dimensional case

The result of an executed command can be modeled by a transition model. This transition model takes the current pose and the action command as input and produces an estimate of the next pose as output. This transition model is essential to the SLAM algorithm to calculate the proposal distribution. In SLAM, the transition model estimates the new pose after a motion command, therefore it is referred to as the motion model.

When the movement of the robot is 1-dimensional, the motion model is determined simply by repeatedly executing different action commands many times and measuring the resulting real translation each time. Assuming the model output is a linear combination of the input with additional Gaussian noise, the parameters of the model can be found by calculating a linear function of the means and the covariances of the outcomes for the different action commands.

Two-dimensional case

When the environment of the robot is planar, the pose of the robot can be described by a location (x and y) and an orientation (angle θ) (see equation 6-12). Where θ is the angle between the orientation of the robot and the direction of the x-axis, see Figure 6-14.

$$pose = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \qquad (6\text{-}12)$$

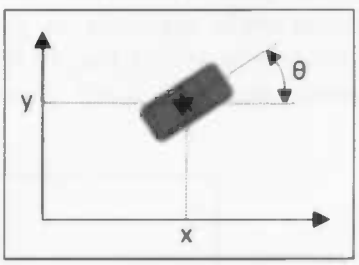

FIGURE 6-14, ROBOT POSE

The motion model depends on the kinematics of the robot. Most robot systems use wheels for means of transportation. Within the class of (indoor) wheeled robotic systems, the majority use differential drive (Ashmore & Barnes, 2002). With differential drive, the robot has two powered wheels with an additional pivotate wheel for balance. The robot can move forward and backward by powering the left and right wheel simultaneously and it can rotate by a difference in the angular speed of the left wheel (ω_l) and the right wheel (ω_r) (Figure 6-15). As a result, the robot has a translational speed (v) and an angular speed (ω).

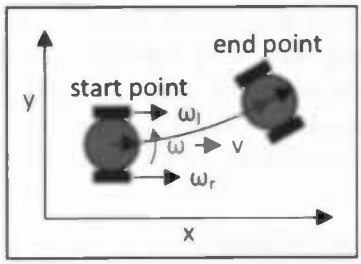

FIGURE 6-15, DIFFERENTIAL DRIVE

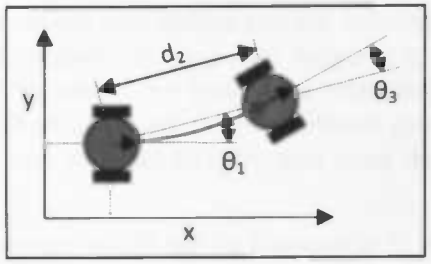

FIGURE 6-16,THREE SUB-STAGES; ROTATION 1 (θ_1), TRANSLATION (d_2), ROTATION 2 (θ_3),

The movement of the robot is illustrated by the curved blue line. The total movement is a function of the translational and angular speed (see equation 6-13). This nonlinear movement must be estimated by the motion model. When the time between two update steps is small enough, the translational speed and the angular speed are approximately constant. This simplifies the problem.

6-13)
$$u_t = \begin{bmatrix} v_t \\ \omega_t \end{bmatrix}$$

The update step can be simplified further by dividing it into three stages (Figure 6-16). First, the robot makes a small rotation (θ_1), then the robot is translated (d) and finally the robot makes a small rotation again (θ_2). The complex movement can now be estimated by three sub-stages that are linear functions, all three adding an amount of Gaussian noise.

Many other traction systems exist besides differential drive. These are for example; omni-drive, synchro drive, tricycle drive and Ackerman steering (Borenstein, Everett, Feng, & Wehe, 1997). With omni-drive, the robot can move in all the directions in the horizontal plane and at the same time rotate around its vertical axis. The robots that use this traction system are often small. Ackerman steering, which is used in the Robojeep, is most common in large robots (see Figure 6-17). An advantage of the traction system compared to the Ackerman steering is that the robot can turn on its spot. On the other hand, with Ackerman steering the angular speed of the car changes more gradually. The angular speed depends on the translational speed and is more under control.

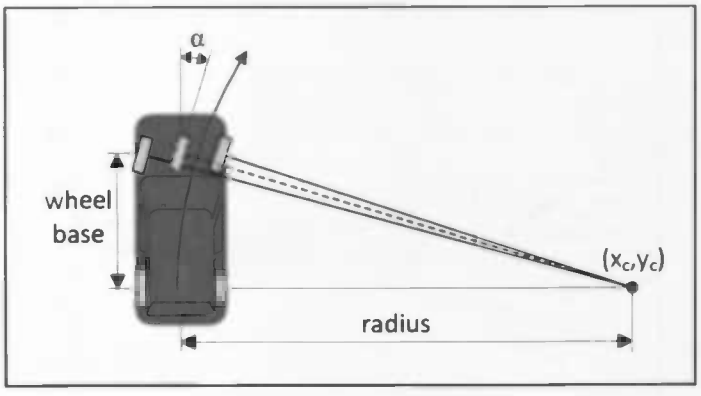

FIGURE 6-17, ACKERMAN STEERING (BORENSTEIN, EVERETT, FENG, & WEHE, 1997)

The path driven by the Robojeep is a function of the rotation of the rear wheels and the steering angle. As illustrated by Figure 6-17, the car drives over an arc with midpoint (x_c, y_c). This midpoint is dependent on the steering angle. When this angle and the speed of the wheels are known, the rotational speed of the car can be calculated. The path can now also be described using only the translational speed and the rotational speed, similar to differential drive. Therefore, the path can be estimated using the same simple sub-stages.

If the robot moves on a flat surface, the estimations of the motion model can be changed to fit the motion of the real robot by adjusting the two translational errors and the rotational error. If, for example, the robot executes an action command of driving 100 meters forward and the rotational error is relatively high, compared to the translational error, then the probability density function can have a shape like the one in Figure 6-18. If on the other hand, the translational error is high compared to the rotational error, then the estimation can have a partitioning such as shown in Figure 6-19. Figure 6-20 shows the result when both the translational error and rotational error are high.

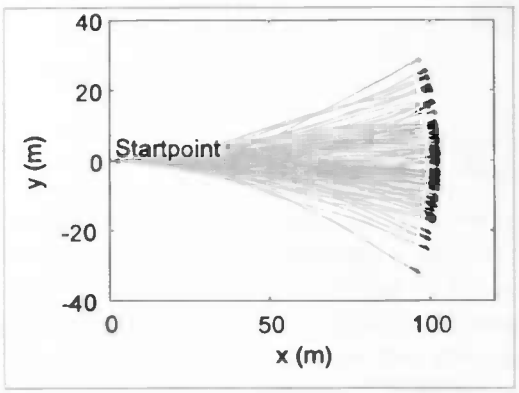

FIGURE 6-18, POSITION ESTIMATION WITH LARGE ROTATIONAL ERROR, FOR 100 METER TRACK

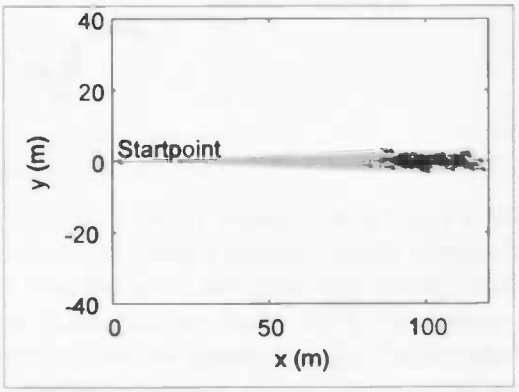

FIGURE 6-19, POSITION ESTIMATION WITH LARGE TRANSLATIONAL ERROR, FOR 100 METER TRACK

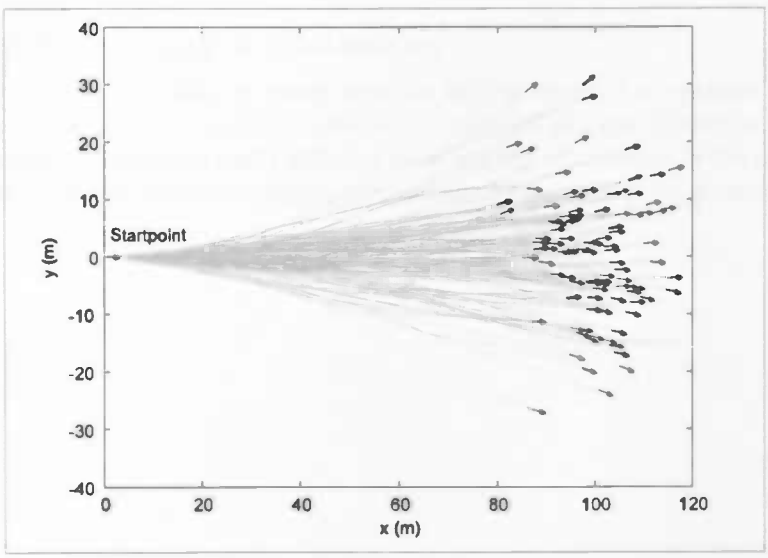

FIGURE 6-20, POSITION ESTIMATION WITH, LARGE ROTATIONAL ERROR AND LARGE TRANSLATIONAL ERROR, FOR 100 METER TRACK

Three dimensional case

If the experiments were all run on the public road, which is approximately flat, the two-dimensional representation could do. However, during some of the tests the Robojeep drives through rough terrain. Translation occurs not only in the horizontal plane but the position of the Robojeep also changes in height. This adds depth as a third dimension to the Cartesian coordination system. The added dimension is the z-axis and because the robot can not only "yaw" but also "pitch" and "roll", the total degree of freedom is 6 (see Figure 6-21).

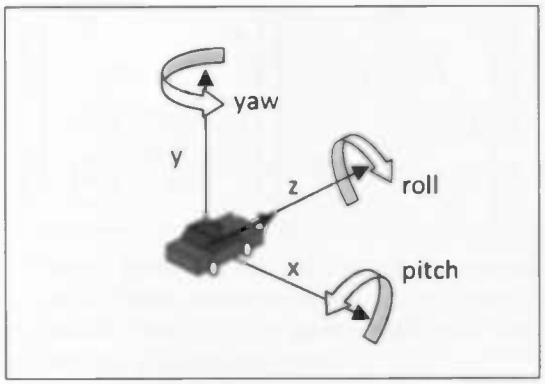

FIGURE 6-21, ROBOT POSE,6 DEGREES OF FREEDOM

Estimating the position of the robot relative to its start position from its wheel rotations, like in the 2-dimensional case, is called odometry. Most SLAM approaches use reliable motion models derived from odometric hardware. A disadvantage of this approach is that the error can grow very quickly. For this experiment, the ego-motion estimation is based on stereo vision (Van der Mark, 2007). This reduces the total ego-motion estimation error and eliminates the error caused by wheel slippage. The ego-motion can be estimated for 6 degrees of freedom. Although a control command always precedes the movement of the robot and the stereo vision based ego-motion can only be estimated afterwards, this estimation will be used as a control command for the SLAM algorithm, to calculate the initial state in the prediction step.

Advantages and disadvantages of odometry

This method uses encoders to measure wheel rotation and/or steering orientation. The advantage of odometry is that only a simple construction is needed to measure angular movement of the wheels. The disadvantage of odometry is that it is easily affected by wheel slip, fluctuation in tire pressure, the timing in between the rotation of the wheels et cetera, additionally the position error grows without bound.

6.5.2 VISUAL ODOMETRY

The translation and rotation of the camera can be estimated, by tracking the features over subsequent frames. SIFT feature descriptors are used to match keypoints between images. Of the SIFT features the 3D coordinates can be calculated. The uncertainty in position can be expressed in a $3 * 3$ covariance matrix. This uncertainty is not the same in each direction. The angle to the SIFT feature can be determined relatively precise compared to the depth, resulting in anisotropic inhomogeneous confidence regions (see Figure 6-22).

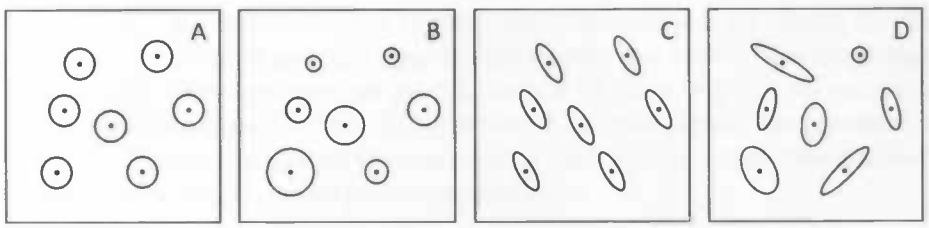

FIGURE 6-22, OF 4 DIFFERENT NOISE MODELS, THE CONFIDENCE REGIONS ARE SHOW; A) ISOTROPIC HOMOGENEOUS, B) ISOTROPIC INHOMOGENEOUS, C) ANISOTROPIC HOMOGENEOUS, D) ANISOTROPIC INHOMOGENEOUS, WHICH IS THE NOISE MODEL THAT HAS TO BE DEALT WITH WHEN ESTIMATING EGO-MOTION USING STEREO VISION.

By assuming that the features are static, the ego-motion can be estimated. The optimal ego-motion is estimated by matching SIFT feature pairs and calculating the rotation and translation that is most likely, gives the positions and covariance matrices of the SIFT features. The precise method is outside the scope of this master's thesis, it can be found in the PhD thesis that is titled: "Stereo and color vision techniques for autonomous vehicle guidance", by Van der Mark (2007).

Vision based odometry has the advantage, compared to odometry based on wheel rotations, that it is not affected by the tire pressure or wheel slip. The error however, still grows cumulative, unless an independent reference is used periodically to reduce the error.

7 VISION ONLY SLAM (σSLAM) USING RAO-BLACKWELLIZED PARTICLE FILTERS

SLAM is a fundamental capability when a robot needs to navigate autonomously in an unknown environment. A problem that arises is that when the robot builds up the map, the error of the landmark location is dependent on the error in the pose of the robot. On the other hand, as the robot needs to localize itself from this map, this location error depends on the error of the landmarks.

Solutions based on a Kalman filter (section 5.1) solve the SLAM problem by storing the dependencies between the position of the robot and the separate landmarks in a matrix. The advantage of this approach is that it gives the optimal solution for the measurements, assuming that the noise is truly normal distributed and the transformation function is truly linear. A disadvantage of this approach is the size of the matrix and the time it takes to update this matrix after each measurement. The size and the update time scale quadraticly with the amount of perceived landmarks.

Beside the Kalman filter and the EKF, a particle filter is also an option to solve the SLAM problem. A particle filter estimates the real state by using a number of particles. The great advantage of this method is that the problem does not have to be linear because the layout of the particles can have all kinds of configurations. In an ambiguous situation, the particle cloud can be split up to examine multiple possibilities.

A disadvantage of the particle filter is that the number of particles needed, can easily grow out of bound when both the pose of the robot and the landmarks are estimated by particles. Rao-Blackwellization (Casella & Robert, 1996) is a technique that can decompose the SLAM problem into two sub problems:

- a localization problem
- landmark estimation problems conditioned on the robot pose estimate

Stereo Vision SLAM, abbreviated by σSLAM, will be used in the experiments in this thesis. It is described in the article "Stereo vision SLAM using the Rao-Blackwellized particle filter and a novel mixture distribution" (Sim, Elinas, Griffin, & Little, 2005). The authors utilize the Rao-Blackwellization technique to solve the SLAM problem based on stereo vision only. The following subsections are based on this article.

7.1 MAP BUILDING

In σSLAM, which is based on a particle filter, each particle represents not only the current pose but also the complete path of the robot until that time step ($s_{0:t}$). The map that is build is dependent on this path. Therefore, each particle keeps track of its own map. A map is a vector of SIFT features which are used as landmarks (see equation 7-1).

7-1) $$map = [l_1\ l_1\ \dots l_n]^T$$

A landmark is a vector as described in equation 7-2.

7-2) $$l = \{P, C^G, \alpha, s, f\}$$ (Sim, Elinas, Griffin, & Little, 2005)

Where:

- $P = [X^G, Y^G, Z^G]$ is the position of the landmark in global map coordinates
- C^G is the $3x3$ covariance matrix for the position of the landmark
- α, s and f are respectively, the angle, the scale and the 128 dimensional descriptor of the SIFT feature as described in subsection 6.3.

7.1.1 TRANSFORMATION OF A LANDMARK POSITION FROM ROBOT TO MAP COORDINATES

The position and the covariance per landmark are measured with respect to the pose of the robot. The values need to be translated from robot coordinates to map coordinates before they can be incorporated into the map.

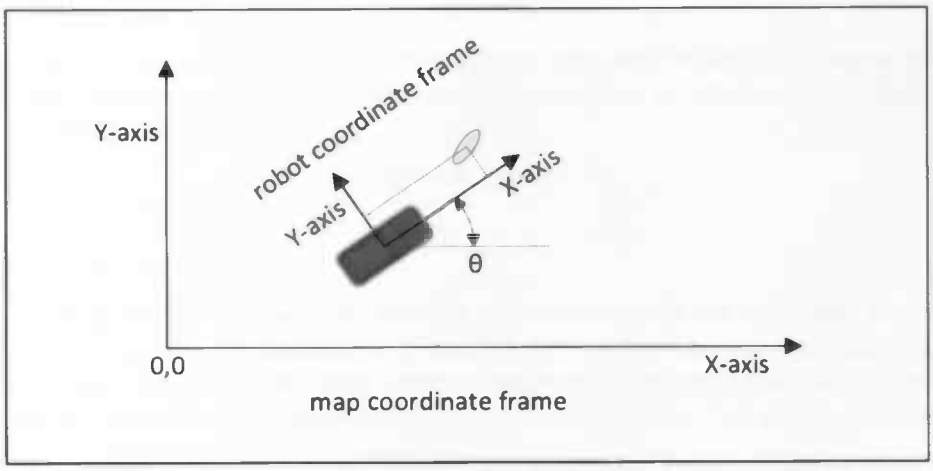

FIGURE 7-1, ROBOT COORDINATE TO MAP COORDINATE CONVERSION

If x_1 is the position of the first landmark in robot coordinates, x'_1 is the landmark position in map coordinates, x'_{robot} is the position of the robot in map coordinates and R is the rotation matrix, then x_1 and x_1' are related by equation 7-3.

7-3) $$x_1' = x'_{robot} + R.x_1$$

In the 2-D case, there is only the rotation of angle θ of the robot. Rotation matrix R is calculated by 7-4.

7-4) $$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Equation 7-3 and 7-4 give:

7-5) $$\begin{bmatrix} x'_1 \\ y'_1 \end{bmatrix} = \begin{bmatrix} x'_{robot} \\ y'_{robot} \end{bmatrix} + \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$$

In the 3-D case, the roll of γ around the x-axis, the yaw of β around the y-axis and the pitch of α around the z-axis are calculated by respectively equation 7-6, 7-7 and 7-8.

7-6) $$R_x(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix}$$

7-7) $$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}$$

7-8) $$R_z(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The total rotation matrix is:

7-9) $$R(\alpha,\beta,\gamma) = R_z(\alpha).R_y(\beta).R_x(\gamma)$$

7.1.2 TRANSFORMATION OF THE POSITIONAL COVARIANCE OF A LANDMARK FROM ROBOT TO MAP COORDINATES

The covariance matrix of a landmark in map coordinates (cov') can be calculated from the covariance matrix in robot coordinates (cov) by equation 7-10.

$$cov' = R.cov.R^{-1} \tag{7-10}$$

For a rotation matrix in an orthogonal basis with a determinant equal to one, the inverse matrix is equal to the transposed matrix therefore; equation 7-11 also holds (Van de Craats, 2003). The transposed is calculated more efficiently.

$$cov' = R.cov.R^{T} \tag{7-11}$$

7.1.3 MAP UPDATE

When the path of the robot is known, the position estimations of all the landmarks are uncorrelated. Therefore, assuming the particle correctly estimates the path of the robot, the landmarks positions can be calculated by uncorrelated Kalman filters. When a landmark is perceived that has not been perceived before, the map can be updated by simply adding the feature descriptor (7-2) to the map vector (7-1).

If a landmark is perceived that is already on the map, the position of that landmark on the map has to be updated by taking the weighted mean (see Figure 7-2). The resulting covariance is the mean of the covariance of the old landmark on the map and the covariance of the landmark given the current measurement in map coordinates.

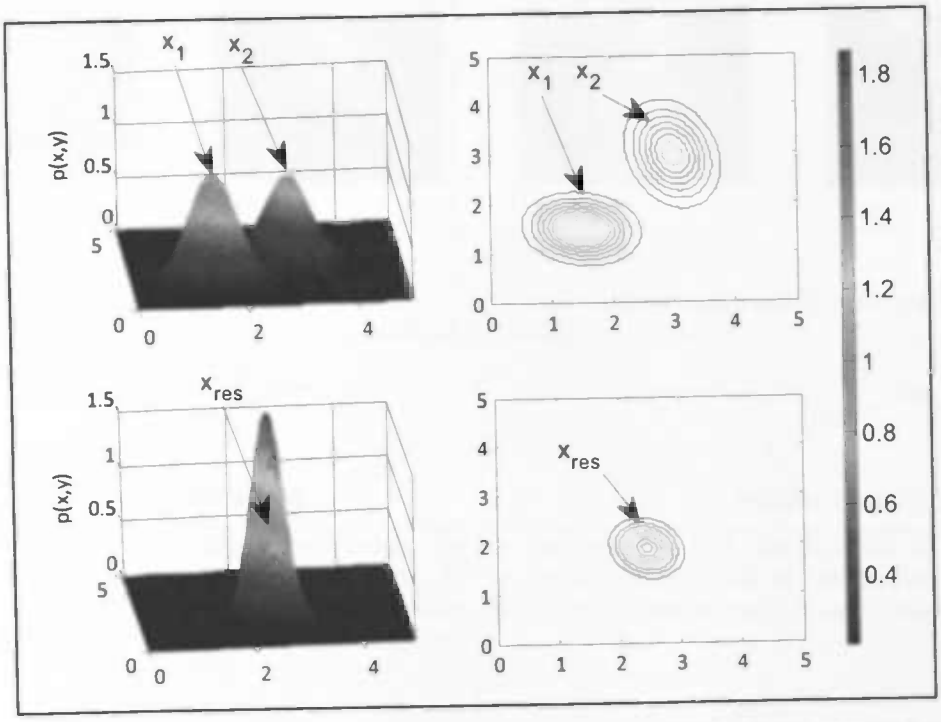

FIGURE 7-2, UPDATING A LANDMARK ON THE MAP. IF x_1 IS THE POSITION OF A LANDMARK ACCORDING TO THE MAP AND x_2 IS THE POSITION ON MAP WHERE IT IS CURRENTLY PERCEIVED, THE VALUE ON THE MAP IS REPLACED BY THE WEIGHTED MEAN (x_{res}). THE RESULTING COVARIANCE IS THE MEAN OF THE COVARIANCE OF x_1 AND x_2.

7.2 MOTION MODEL

When the positions of the landmarks compared to the robot at each time step are known and so is the data association, the displacement of the Robojeep can be estimated by tracking the static landmarks over time (Figure 7-3).

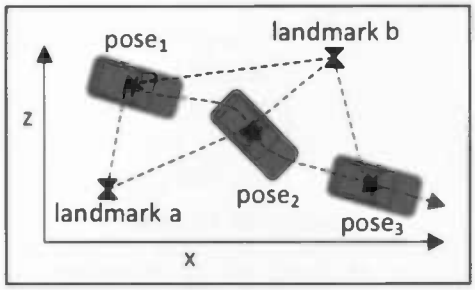

FIGURE 7-3, EGO-MOTION ESTIMATION. ASSUMING THE ENVIRONMENT IS STATIC; THE MOVEMENT CAN BE ESTIMATED BY TRACKING THE LANDMARKS (a AND b) IN THE ENVIRONMENT EACH TIME STEP. THE VEHICLE CANNOT PERCEIVE THE LANDMARK IN POSE 3.

The positions of the landmarks as seen from the robot are obtained by stereo vision (see Figure 7-4). How this can be done was discussed in more detail earlier on, in section 6.4. Pairs of 3D points can be found by matching successive landmarks sets. These pairs can be used to estimate the motion of the camera of the robot (see Figure 7-5).

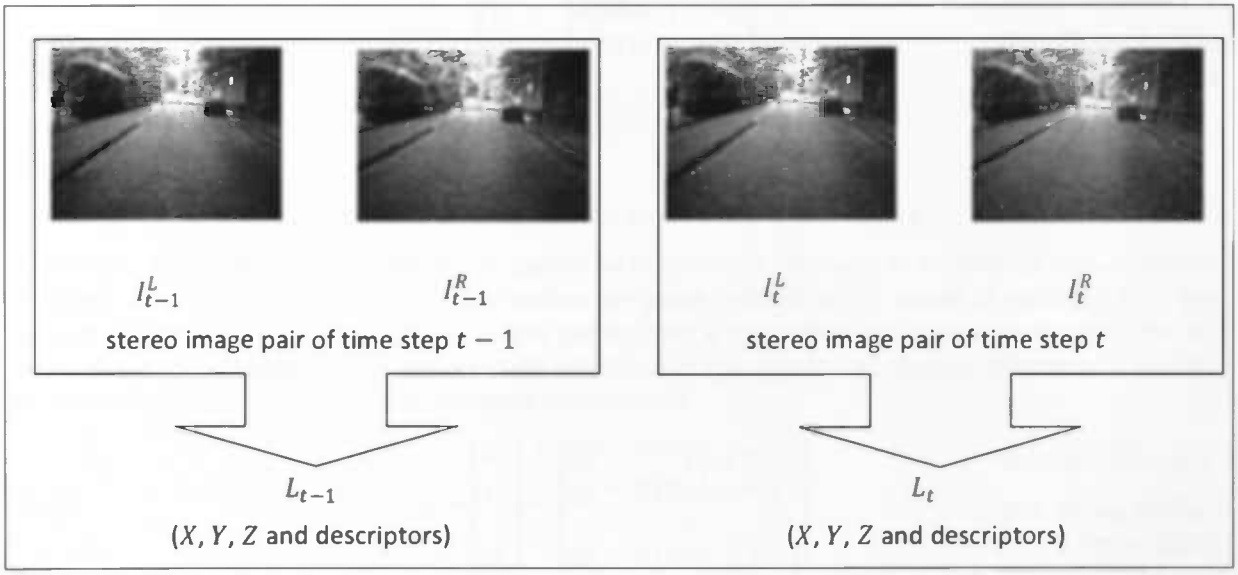

FIGURE 7-4. FOR EACH STEREO IMAGE PAIR PER TIME STEP THE SIFT FEATURES CAN BE MATCHED. THE MATCHING FEATURES ARE USED AS LANDMARKS, RESULTING IN TWO SETS OF LANDMARKS (L_{t-1} AND L_t) FOR RESPECTIVELY TIME STEP $t-1$ AND t. THE X, Y AND Z VALUES OF THESE LANDMARKS CAN BE CALCULATED.

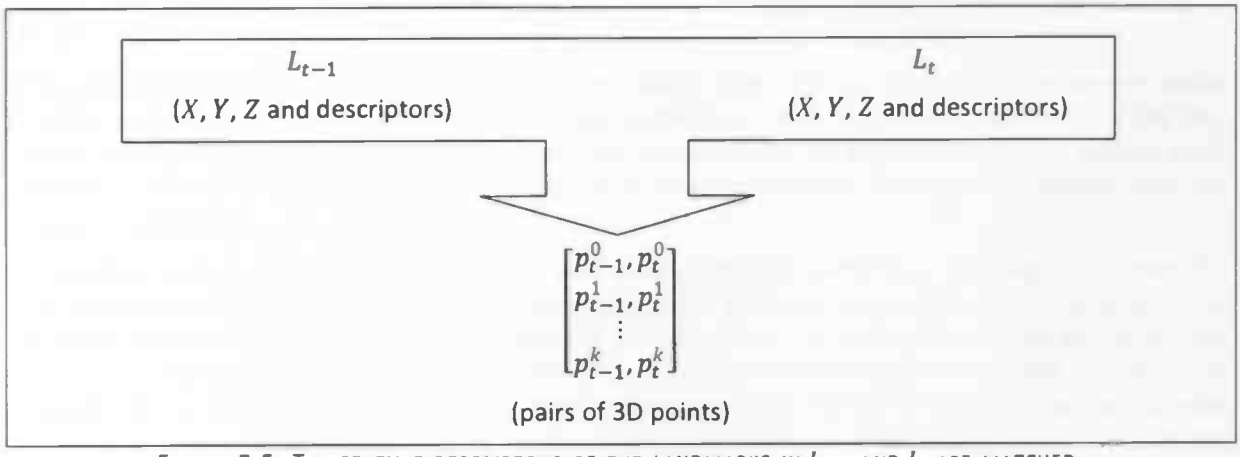

FIGURE 7-5, THE FEATURE DESCRIPTORS OF THE LANDMARKS IN L_{t-1} AND L_t ARE MATCHED. OF EACH MATCHING LANDMARK, THE X, Y AND Z POSITION AT TIME STEP $t-1$ AND TIME STEP t IS KNOW, RESULTING IN A SET OF PAIRS OF 3D POINTS (p_{t-1} AND p_t). BY USING THESE, THE DISPLACEMENT OF THE ROBOT CAN BE CALCULATED BY AN ITERATIVE CLOSEST POINT ALGORITHM (DIOSI & KLEEMAN, 2005) OR (VAN DER MARK, 2007).

The movement of the camera, each time step can be decomposed into a three dimensional translation (T) and a three dimensional rotation (R). The points of a point pair originate from the same landmark so equation 7-12 will hold.

7-12)
$$\begin{bmatrix} p_t^0 \\ p_t^1 \\ \vdots \\ p_t^k \end{bmatrix} = \begin{bmatrix} K(Rp_{t-1}^0 + T) \\ K(Rp_{t-1}^1 + T) \\ \vdots \\ K\left(Rp_{t-1}^k + T\right) \end{bmatrix}$$
(Sim, Elinas, Griffin, & Little, 2005)

Where:

- K is a calibration matrix that depends on the parameters of the camera

In practice, the positions of the landmarks cannot be determined precisely. In addition to this uncertainty, there can also be false matches. False matches can cause outliers, which should be excluded from the motion estimation. The distance between the points after a translation and a rotation is called the re-projection error (equation 7-13). For a certain rotation and translation, the re-projection error is minimal. How this can be calculated is not discussed in this thesis.

7-13)
$$\varepsilon = \begin{bmatrix} \varepsilon_0^T \\ \varepsilon_1^T \\ \vdots \\ \varepsilon_k^T \end{bmatrix} = \begin{bmatrix} p_t^0 - K(Rp_{t-1}^0 + T) \\ p_t^1 - K(Rp_{t-1}^1 + T) \\ \vdots \\ p_t^k - K\left(Rp_{t-1}^k + T\right) \end{bmatrix}$$
Re-projection error
(Sim, Elinas, Griffin, & Little, 2005)

Occasionally, the estimation of the ego-motion estimator is completely wrong. This can happen in a sharp corner, for example, when too few matching landmarks are perceived in sequential frames. If the environment is not static but the main part of the image is filled with a moving car, for example, this can also influence the estimation. An estimation that is physically not possible, for example when the robot is moved up side down in a split second can be filtered out. An additional sensor, like an IMU, could also be used to narrow the search space for the motion estimator.

7.3 SENSOR MODEL

The movement of the robot makes the overall uncertainty grow. On the other hand, the sensor model can be used to reduce this uncertainty. The sensor readings are used to calculate the fitness of the particles. This fitness is called weight and it is used when new particles are drawn from the old particle pool. Particles with more weight have a higher chance of being re-sampled. This section explains how the weight is calculated.

In addition to the pose of the robot, each particle represents a complete map. The map of the particles can be different because of the different trajectories of the particles. As mentioned, the weights of the particles need to be calculated before re-sampling can take place. To do this, the current perceived features are compared to the map of each particle. If there is a good fit, the weight is high and vice versa. To calculate the distance between a perceived feature and a matching feature on the map the mahalanobis distance is used, which was discovered by the Indian scientist P. C. Mahalanobis (1856-1922). It expresses the distance between variables taking covariance among the variables into account.

To determine where a certain landmark is in the environment, by using the information from the sensors, the landmarks need to be identified first. This is referred to as the correspondence problem introduced in subsection 5.1.2. Identification means that the robot knows, to which landmark on the map a perceived landmark corresponds, or that it is a new landmark when it is not perceived before. An observation is the set of correspondences between landmarks on the map and in the current observation (equation 7-14).

7-14) $$z_t = \bigcup\nolimits_{1\dots k} \{l_i \leftrightarrow o_j\}, i \in [1 \dots m], j \in [1 \dots n]$$ (Sim, Elinas, Griffin, & Little, 2005)

Where:

- l is a landmark on the map
- o is a landmark in the current observation
- k is the number of correspondences between landmarks in the map and the current observation
- m is the number of landmarks in the map
- n is the number of landmarks in the current observation

The weight of a particle can be calculated by taking the log-likelihood of its observations. The observation log-likelihood can be calculated by taking the sum of the feature correspondences (equation 7-15).

7-15) $$\log p(z_t|m_t^i) = \sum_k \log p(o_k|l_k^i)$$ (Sim, Elinas, Griffin, & Little, 2005)

The feature correspondent is given by equation 7-16.

7-16) $$\log p(o_k|l_k^i) = -0.5min\left(T_l, (P_{o_k}^G - P_k^G)^T S^{-1}(P_{o_k}^G - P_k^G)\right)$$ (Sim, Elinas, Griffin, & Little, 2005)

Where:

- T_l is the maximum observation innovation, to prevent outliers from having too much influence.
- S is the correspondence covariance, see equation 7-17.

7-17) $$S = R_{s_t} C_{o_k}^L R_{s_t}^T + C_k^G$$

Where C_k^G is the landmark covariance and $R_{s_t} C_{o_k}^L R_{s_t}^T$ is the transformed observation covariance ($C_{o_k}^L$). The transformation is discussed in section 7.1.

PART II: CONTRIBUTIONS

8 THE OUTDOOR SLAM IMPLEMENTATION

This chapter covers the approach for using σSLAM in an outdoor situation. First, the research direction will be outlined. The SLAM algorithm, used in this experiment, is implemented in Matlab[4]. In section 8.2, the overall structure and the functional block diagram is outlined, which is very similar to the σSLAM algorithm described in the paper of Sim et al. (2005). The data block is outlined in section 8.3 and the process blocks are outlined in sections 8.4, 8.5 and 8.6. This is where the adjustments are made to the σSLAM algorithm.

8.1 RESEARCH DIRECTION

In this experiment, a vision only solution for a SLAM problem is tested. The landmarks that will be used are naturally occurring landmarks. These landmarks are detected by calculating the SIFT features of images taken by a stereo camera. From the SIFT features of the image of the left camera that have a matching SIFT feature in the image of the right camera during one time step, the position with respect to the camera can be calculated. If all features, of which the position is known with respect to the camera, are stored, the amount of landmarks on the map can grow rapidly. If n is the number of landmarks that are currently perceived, p is the number of particles and m is the amount of landmarks on the map, the complexity of matching the landmarks to the map is, at worst, in the order of $O(n * m * p)$. The research will be focused on both the selection and the storage of the features that are used for mapping.

Matching features from both the left and right image during a time step are matched between successive frames. Some of the features that also match between successive frames are used for the ego-motion estimation. The estimated motion will depend on the features that are used during a time step because of noise and wrong matches. The ego-motion estimator takes the motion that fits the majority of the landmarks. All the features that do agree with the estimated motion are returned by the ego-motion estimator, the others are deleted.

The features that are deleted are considered outliers and therefore thrown out. This reduces the amount of errors and the amount of features, which makes the mapping step less complex. Still, many features remain and this may be too many to handle if the route of the robot becomes longer or if the time to process the information is limited. Other selection criteria, which are used in this experiment, are the height of the feature and the distance of the feature with respect to the sensor. Additional to these experiments two methods are compared for accepting or rejecting a match depending on the distance between the descriptor of the landmarks.

[4] Matlab®: http://www.mathworks.com/

8.2 STRUCTURE

The global functional diagram (Figure 8-1) shows the global structure of the SLAM algorithm, which is processed in a loop. Starting at the data block "Map", after receiving new images, the new received features can be matched to the features of the previous time step. If the Robojeep is moving, the matching features are translated and/or rotated slightly compared to the previous time step. The route of the Robojeep can be deduced from all these small translations and/or rotations. By this information, the data block "Position and path" can be updated. When the position is known, the features from the current stereo image that were used to deduce the translation and rotation from, can be used to update the map. The current features can also be compared to the map to detect a loop closure. This brings us back at the data block "Map". When new images are available, the process starts over again.

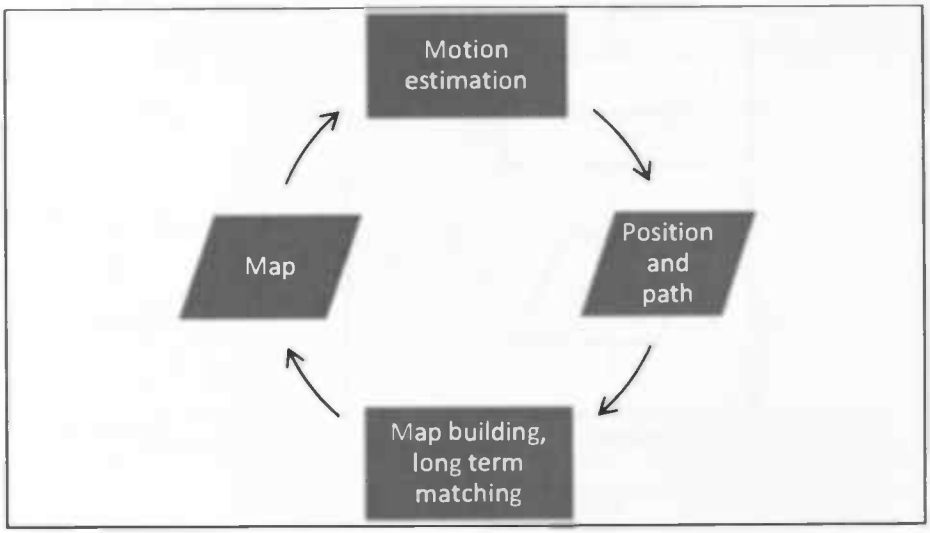

FIGURE 8-1, GLOBAL FUNCTIONAL DIAGRAM OF THE SLAM ALGORITHM

A more detailed version of the functional block diagram of the software test model is presented in Figure 8-2. The σSLAM part is situated on the right of the figure, in thick black lines. The data blocks in σSLAM are the particle poses and maps (section 8.3). The process blocks of σSLAM are; the motion model (section 8.4), the sensor model (section 8.5) and the map update (section 8.6).

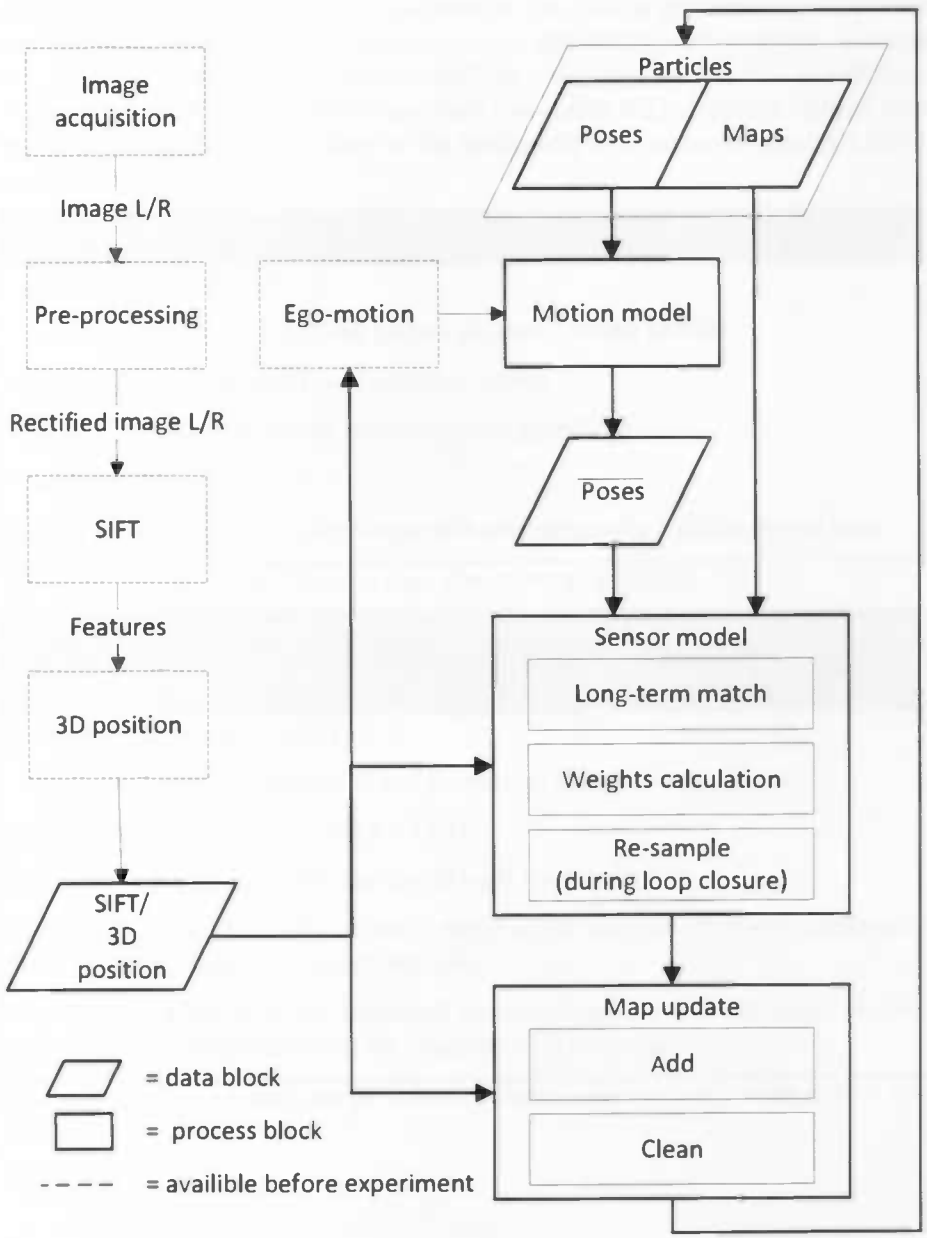

FIGURE 8-2, THE FUNCTIONAL BLOCK DIAGRAM OF THE σSLAM MATLAB IMPLEMENTATION

8.3 DATA BLOCKS

Each particle holds the complete track of the vehicle and stores a map (see Table 8-1). What is stored on the map depends on the settings of the implementation. The data structure "local stored map" of each particle contains at least the position and the covariance of each landmark on that map. The rest of the content depends on the option chosen in the parameter file. This is explained in the following sections about the process blocks. Here it is assumed that two storage settings are available. If option "local storage" is chosen, the data structure "local stored map" for each particle also contains the descriptor, the initialization moment and the score of each landmark (see Table 8-2). If option "global storage" is chosen, all the variables, except for the position of the landmarks and the covariance, are stored only once on a global stored map.

A particle consists of:
The particle position $[3 * 1]$
The route , which is a list of the former positions of the particle
The particle orientation $[3 * 3]$ rotation matrix
The covariance matrix of the position of the particle$[3 * 3]$
The weight of the particle
A local stored landmark map per particle and optionally a global stored map

TABLE 8-1, THE DATA STRUCTURE OF A PARTICLE

Option 1: There is no global stored map, only local stored maps. Each local stored map consists of:
A list of landmark positions $[3 * 1]$
A list of covariances matrices of the position of the landmarks $[3 * 3]$
A list of landmark descriptors $[1 * 128]$
A list of the moment of initialization of each landmark
A list of the score of each landmark, which is the number of times a landmark has been perceived after its initialization
A list of variables, one per landmark on the map, to store the index of the matching new landmark (used for programming purposes)

TABLE 8-2, THE DATA STRUCTURE OF A LOCAL STORED LANDMARK MAP, WHEN OPTION 1 IS CHOSEN IN THE PARAMETER FILE

Option 2: There are local stored maps and one global stored map.
Each local stored map consists of:
A list of landmark positions $[3 * 1]$
A list of covariances matrices of the position of the landmarks $[3 * 3]$
The global stored map consist of:
A list of landmark descriptors $[1 * 128]$
A list of the moment of initialization of each landmark
A list of the score of each landmark, which is the number of times a landmark has been perceived after its initialization
A list of variables, one per landmark on the map, to store the index of the matching new landmark (used for programming purposes)

TABLE 8-3, THE DATA STRUCTURE OF EACH LOCAL MAP, STORED PER PARTICLE AND THE STRUCTURE OF THE GLOBAL MAP, WHEN OPTION 2 IS CHOSEN IN THE PARAMETER FILE

8.4 MOTION MODEL

In the motion model block, the estimated pose is calculated from the previous pose. The input variables for this block are the translation and rotation at the current time step calculated by the motion estimator. For every particle, at every time step, a translational error and a rotational error is added to the stereo vision based ego-motion estimation. The translational and rotational errors are drawn from normal distributions. The variances of these normal distributions are important settings. They have to be large enough to capture the error introduced by the motion estimator, but not too large otherwise positional information is lost.

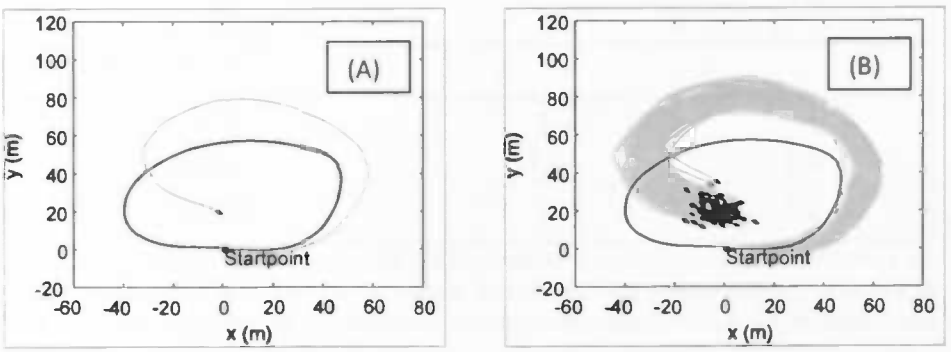

FIGURE 8-3 (A), PART OF THE DATA OF A TEST RUN. THE SINGLE BLUE LINE IS THE GPS DATA, WHICH SHOWS THAT THE ROBOJEEP CLOSES A LOOP DURING THIS RUN. THE GRAY LINE, WITH AN ARROW, SHOWS A RANDOM ESTIMATED ROUTE OF THE ROBOJEEP CALCULATED BY THE MOTION ESTIMATOR. FOR NOW, THE EGO-MOTION ESTIMATOR IS ASSUMED TO HAVE NO BIASED ERROR. THERE MIGHT BE A BIAS, BUT THIS WILL BE INVESTIGATED SEPARATE. FIGURE 8-3 (B) SHOW THE SAME RUN BUT NOW SOME TRANSLATIONAL AND ROTATIONAL NOISE IS ADDED TO THE MOTION ESTIMATE. THE NUMBER OF PARTICLES USED HERE IS 100. SOME OF THE PARTICLES ARE A BETTER GUESS OF THE POSITION OF THE ROBOT THEN THE EGO-MOTION ESTIMATE WITHOUT THE ADDED NOISE. THE AMOUNT OF NOISE IS TOO LOW BECAUSE THE CLOSEST PARTICLE HAS NOT REACHED THE CORRECT ENDPOINT CLOSE ENOUGH.

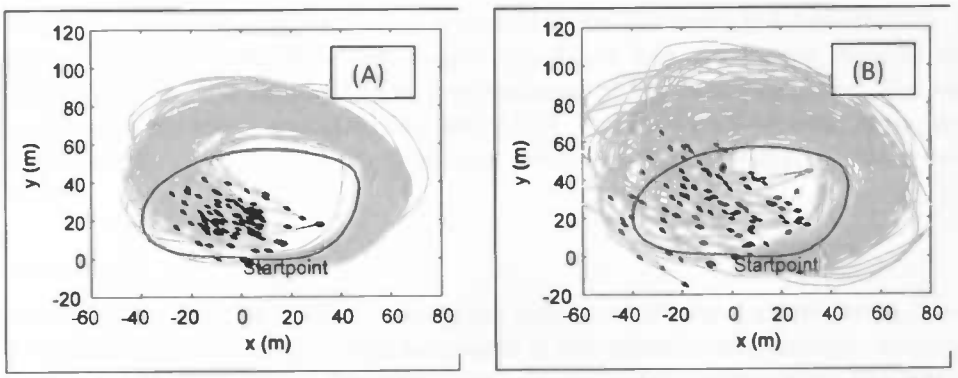

FIGURE 8-4 (A), THE TRANSLATIONAL AND ROTATIONAL VARIANCES OF THE ERROR ARE TWICE THE VARIANCES OF THE ERRORS USED IN FIGURE 8-3 B. THE CLOUD OF PARTICLES HAS GROWN AND NOW THE CLOSEST PARTICLES ARE FAR BETTER GUESSES THEN THE MOTION ESTIMATION WITHOUT NOISE ADDED TO IT. WHEN THE VARIANCES OF THE ERRORS ARE DOUBLE ANOTHER TIME, THE PARTICLE CLOUD DIFFUSES EVEN MORE (FIGURE 8-4 B). IN THIS FIGURE, THE POSITIONAL INFORMATION ESTIMATED BY THE STEREO VISION EGO-MOTION ESTIMATOR HAS DEGRADED FURTHER.

When the amount of noise that is added is too large, too much information is lost because the particles spread out too much. However, estimating the error a bit too low is worse. Then, changes are that the true pose is not estimated correctly by any of the particles. Therefore, setting the right parameters is not a real delicate job as long as the parameters are not underestimated (Thrun, Burgard, & Fox, 2005).

Error variance	2D ego-motion	3D ego-motion
Translation (mm)		
X	1	1
Y	1	1
Z	–	10
Rotation ($degrees$)		
$Roll$	–	1.1
Yaw	1.1	1.1
$Pitch$	–	7.2

TABLE 8-4, THE SETTING USED FOR THE MOTION MODEL DURING THE EXPERIMENTS. IT IS ASSUMED THAT THERE IS NO BIAS IN THE ERROR. THEREFORE, THE MEANS ARE 0. THE ERROR IN THE PITCH OF THE MOTION ESTIMATION PROVED, AFTER SOME TRIALS, TO BE RELATIVELY LARGE.

8.5 SENSOR MODEL

Vision is a passive measurement because no signal is transmitted by the sensor. In contrast, for example, laser, ultrasonic and most sonar measure devices are active. A passive sensor is an advantage over an active one when it is important for the robot to remain undetected (Van der Mark, 2007). This is often the case when the sensor is used for military purposes. A stereo camera is used in this experiment because its advantage, stereopsis, does weight up to its disadvantage, the small viewing angle compared to a camera with a wide angle lens or an omni-vision camera.

The theory about the sensor model used in the implementation is treated in subsection 7.3. The sensor model consists of the sub blocks; matching, weight calculation and re-sampling. Specific choices that have to be made for matching the landmarks are described in the next subsection. The matches are weighted by means of the mahalanobis (subsection 8.5.2). The particles are re-sampled depending on the weight of the particles. How this is done is also not trivial. The choices that are made are discussed in subsection 8.5.3.

8.5.1 MATCHING

Perceived landmarks are matched to landmarks on the map to determine if a landmark is seen before or if it is a new landmark. The similarity of two landmarks is calculated by the Euclidian distance between their 128 dimensional SIFT descriptors. The landmarks are considered a match, if the distance is less than a certain threshold. For the experiments, the matching threshold can be applied in two ways. The threshold can have an absolute value, or the threshold can have a relative value, where it depends on second best matching distance. In the later case, the current perceived landmark and its best matching landmark on the map, are called a match if the distance is less than a certain percentage of the distance between the current perceived landmark and the second best matching landmark on the map. Relative matching is used in the original σSLAM algorithm by SIM et al. (2005). It is also used by the ego-motion estimator.

Matching threshold	Value
Absolute	0.25
Relative	$0.6 * distance_{second\_best}$

TABLE 8-5, MATCHING THRESHOLD USED FOR EVALUATING DESCRIPTOR DISTANCES

Search area

If the search space, for matching new landmarks to the landmarks on the map, is the complete map, matching can take a lot of processing time, especially if the map is large. In a particle filter, each particle indicates the position of the robot. By using this robot position and knowing the relative position of the landmarks compared to the robot, it is possible to calculate the position of the new landmark on the map. When the position is known, the positional distance between the landmarks on the map can be calculated. Determining the positional distance, in two-dimensional space if the height is neglected, is much faster than determining the descriptor distance in 128-dimensional space.

It may be the case that landmarks on the map, that are far too distant, do not have to be matched based on their descriptors to the new perceived landmarks. Therefore criterion 1A (Table 8-6) is used as a pre-selection. This is only a rough pre-selection because the landmark covariance is not taken into account. The maximal distance threshold must therefore be set high. On the other hand, it is fast and it can reduce the search space substantially and may therefore be efficient to use. The amount of reduction of the search space will be investigated.

Selection criteria for matching landmarks				
1A	Difference in x and y position	+	1B	Descriptor distance
2	Descriptor distance			

TABLE 8-6, THE SELECTION CRITERIA, FOR MATCHING NEW LANDMARKS TO LANDMARKS ON THE MAP

8.5.2 WEIGHT CALCULATION

The weight of a particle can be calculated by taking the log-likelihood of its observations. This can be calculated by means of the mahalanobis distance as described in subsection 7.3. The landmark correspondence is restated here for convenience:

8-1)
$$\log p(o_k|l_k^i) = -0.5min\left(T_l, \left(P_{o_k}^G - P_k^G\right)^T S^{-1}\left(P_{o_k}^G - P_k^G\right)\right)$$
(Sim, Elinas, Griffin, & Little, 2005)

If a match is an outlier, the mahalanobis distance becomes relatively high and will have too much influence on the outcome of the total particle weight. Therefore, a threshold is used (T_l). If the mahalanobis distance is above the threshold value, this threshold value will be used (see Table 8-7).

Maximum observation innovation	Value
T_l	4

TABLE 8-7, A THRESHOLD TO CUT OFF THE MAHALANOBIS DISTANCE ABOVE T_l

8.5.3 RESAMPLE

Landmarks that are currently perceived can be used to localize the robot with respect to the landmarks on the map. The matches are used to calculate the weights of the particles. By using the weights, the particles can be re-sampled. The disadvantage of re-sampling is the problem of coalescence, which was discussed in subsection 5.4.2. Therefore, the re-sampling must not take place too often. An option is to re-sample at a certain interval. Another option is to re-sample when the variance in the weight population is high. If the variance is high, the difference in weight of the particles differs substantially and re-sampling is worthwhile.

The trigger to re-sample, that is used in this experiment, is the detection of a loop. This is when the robot perceives landmarks that where initialized a while ago, when the position of the robot was still less uncertain. To reduce calculations, the list of matching landmarks that was calculated during the forgoing mapping, will be used. For each landmark on the maps that matched to the currently perceived landmarks, the initialization moment is retrieved. If this moment was more than a certain amount of time steps ago, it is called a long-term match. If the number of long-term matches is above a certain threshold, it is considered a loop and re-sampling takes place. The resample method used, is stratified re-sampling (Douc, Cappe, & Moulines, 2005).

Loop closure parameters	Value
Minimal age of a landmark before it is used for long-term matching	40 time steps
Minimal number of long term matches to detect a loop closure	4
Re-sampling method	Stratified re-sampling

TABLE 8-8, SETTINGS FOR DETECTING A LOOP AND TRIGGER RE-SAMPLING

8.6 MAP UPDATE

In the original setting of the vision based SLAM algorithm (Sim, Elinas, Griffin, & Little, 2005), each particle keeps track of its own local map. To increase the speed of the algorithm some of the information of the landmarks on the maps can be stored globally during part of the experiment. The following subsections are applicable to both possible data-blocks, which are shown in Table 8-2 and Table 8-3.

8.6.1 ADDING LANDMARKS

In the map-update-block, the perceived features are added to the map as landmarks. If a perceived landmark had a matching landmark on the map, both their positions and their positional covariances are combined. If there is no match, a new landmark is initialized on the map. Of each landmark on the map, the moment of initialization is stored. In addition, the number of times it is perceived, which is called "the score", is stored per landmark.

8.6.2 MAP CLEANING

The score, together with the initialization moment of each landmark, can be used to clean the map at certain intervals. In part of the experiment, landmarks that are not seen frequent enough in the first period after their initialization are removed from the map.

Map cleaning parameters	Default value
Clean interval	1 ($9999 = no\ cleaning$)
Minimal score	3
Wait time before cleaning = initialization moment – current time step	10

TABLE 8-9, MAP CLEANING SETTINGS

8.6.3 FEATURE SELECTION

The SLAM algorithm used in this experiment is computationally expensive. To limit the time it takes to update after each observation the total amount of landmarks on the map must be limited. The calculation time for each observed landmark grows linear with the amount of landmarks on the map. Deleting uncertain landmarks which are already on the map is one method to reduce the number of landmarks on the map. Another method is features selection. Of a perceived feature, the position of the corresponding landmark can be calculated. This positional information can be used to divide the features into ground plane landmarks and landmarks above the ground plane. Threshold that are used by the algorithm to decide which part is the ground plane are set in the parameter file.

Parameters for selecting ground plane
$(X < -500)\ AND\ (Z < 25000)$

TABLE 8-10, GROUND PLANE SELECTION

9 EXPERIMENTS

The main purpose of the experiments is testing how SLAM can benefit from object recognition. The assumption is that by means of object recognition, a robot can detect if it is at, or near a position it has been before and that loop closure can be used to reduce the uncertainty in position. A Vision only SLAM algorithm will be used in an outdoor setting, where no adjustments are made to the environment.

The paper about the σSLAM (Sim, Elinas, Griffin, & Little, 2005), used indoor, is the starting point of the implementation that is made to be able to perform the tests. During this implementation, some choices have to be made, which are not trivial. This is described in the previous chapter. Difficulties arise for example, because of the large amount of image features that are perceived during test runs. The performance and properties of the implementation are tested for two test runs en several parameter settings.

9.1 RESEARCH QUESTION 1

Experiment 1 and 2 are performed to answer the first research question: Is it possible to use a vision only SLAM approach in a large-scale outdoor environment?

9.1.1 EXPERIMENT 1: LOOP CLOSURE

In the experiments, a robot has to build a map and simultaneously track its position using only stereo vision data. The focus is on loop closure. The first experiment is therefore to test the performance of the algorithm during a loop closure. The route is a small section of the city trip, where the robot is assumed to move in a plane (for clarity: the landmarks on the map have a 3D position). A pool of 100 particles will be used and the performance is measured by the Euclidian distance between the particles and the closest GPS point. If this experiment is a success, the following experiments can also be performed, to improve and further test the performance.

9.1.2 EXPERIMENT 2: PLANAR VERSUS 3D ROBOT MOTION

After the loop closure for planar motion in the previous experiment, the performance of the algorithm is tested for 3D motion. The area at the Waalsdorpervlakte contains small hills, so the complexity of the motion estimation is increased from 3 to 6 degrees of freedom because the height, pitch and roll of the robot is also variable. As a test route, a small circular path is used containing a rather steep hill of about 1.5 meter in height. The performance can be measured by looking at the estimated beginning and endpoint of the route, which should intersect.

9.2 Research question 2

Experiment 3a, 3b, and 4 are performed to answer the second research question: What kind of landmarks can be used best and how can they be matched?

9.2.1 Experiment 3: Deleting uncertain landmarks

3A: Landmarks on and above the ground plane

If a person has to balance on one leg, the person can look at a, not too distant, spot on the ground too use as a reference point. By doing this, the brain can quickly notices a movement of the body and corrects it. If the spot is not too far away, the person is capable of detecting changes in distance to the point and this increases the stability. On the other hand, if a person is lost in a city, he or she will probably not look at the ground to search for something familiar. The person will look at the larger objects in the environment. A tower is an interesting object, for example. It can be perceived from great distance and is distinguishable from other objects. The ground on the other hand, is mostly paved with mud, grass or asphalt that probably looks the same everywhere.

The third experiment is based on the idea that, for localization purposes, the interesting natural landmarks are above the ground plane. Apart from being more distinguishable, landmarks that rise above the ground can often be seen from various distances, which make them more valuable. In the experiments, it is tested if it is possible to reduce the amount of data on the map, by leaving out the landmarks on the ground plane and still perform good loop closure detection to improve the localization and mapping.

3B: Landmark lifetime

To reduce the amount of stored landmarks further, only the landmarks that are seen often enough remain on the map. After 10 time steps, starting at the moment of initialization of the landmark, the score is evaluated. The score is the amount of times a landmark was perceived. If it is below a certain threshold, the landmarks are deleted from the map. The number of landmarks on the map is plotted for threshold set at 1 (all the landmarks remain), 2, 3, 4, and 5. The thresholds are tested once using all the landmarks and once using only the landmarks above the ground plane. The resulting map sizes will be compared

9.2.2 Experiment 4: Absolute versus relative matching distance

At the Waalsdorpervlakte, a relative descriptor distance threshold is used for matching. This does not work as well for the city trip, because the amount of landmarks on the map is too large near the end of the longer trip. Using an absolute descriptor distance threshold may be the solution. Both the thresholds will be tested and the results are compared.

9.3 Research question 3

Experiment 5 and 6 are performed to answer the third research question: How can the efficiency of the vision only SLAM algorithm be increased?

9.3.1 Experiment 5: Pre-selecting landmarks for matching

Is it profitable to pre-select the landmarks based on their position on the map? This is a relevant question if the number of landmarks on the map is large and processing time becomes an issue. The pre-selection criterion 1A, discussed in subsection 8.5.1, will be used. The amount of reduction of the search space is investigated.

9.3.2 Experiment 6: Landmark descriptor stored local or global

Another way to deal with the growing amount of landmarks on the map, is to adjust the algorithm in such a way it can handle more landmarks. Each particle keeps track of its own landmark. If a new landmark is perceived it can be added to the map of each particle. If there are p particles, for each perceived new landmark, in total p new landmarks are added to the maps, where each added landmark consists of a $[3 * 1]$ position vector, a $[3 * 3]$ covariance matrix and a $[1 * 128]$ descriptor vector. Additionally, the score and the initialization moment of each landmark on the map, are stored.

The storage of the landmarks will hardly be a problem and if it is, adding some additionally random access memory to the internal memory of the computer or using a hard disk, will do the trick. However, the extra-added landmarks do also increase the search space and this will easily become a problem, if the amount of landmarks grows too large. If each map has about the same amount of landmarks, then matching n perceived landmarks to a map with m landmarks for each of the p particles, has a total complexity in the order of $O(n * m * p)$.

If there is no need to do multiple hypotheses tracking for the matching of the landmarks because of the distinguish-ability of the 128 dimensional SIFT descriptor, the search space for p particles can be reduced to the order $O(n * m)$. If a pair is matched that should not be matched, or a pair is not matched that should be matched, changes are that this will influence the map of each particle evenly bad and therefore the weights of the particles with respect to each other, stays the same. The landmarks can be stored in one list and after matching, the indexes of the landmarks in the list are used to update the map of each particle. This speedup is also part of the experiment. The advantages and disadvantages will be discussed.

9.4 EXPERIMENTAL SETUP

The test runs are performed in two areas (Figure 3-3 at page 12). One is an urban area in The Hague, the other area is a natural military test terrain near the TNO location, called The Waalsdorpervlakte. During the city trip in The Hague, the Robojeep is assumed to move in a flat plane. The ego-motion has three degrees of freedom (DOF); two for translation and one for rotation. The route at the Waalsdorpervlakte contains small hills. The degree of freedom of the ego-motion will be three for translation and three for rotation. The total test route will contain several loops. Small ones where the traveled distance will be about 100 meters between the start and the closure of the loop and larger ones where the car travels more than 1000 meters before closing the loop. During the test runs there are no dynamical features; the environment is assumed static.

The Robojeep at TNO is used as experimental acquisition platform (Figure 9-1). It has a stereo camera upfront and a GPS receiver on top. The synchronized stereo and GPS data is stored for off line computation. Additional, sensor readings from the odometry and the IMU are stored, to make the data set even more valuable for future research projects. Although the Robojeep is able to drive autonomously, it will be driven manually. The driving speed on natural terrain is 5-25 km/h. In the urban area, the speed will be up to 50 km/h.

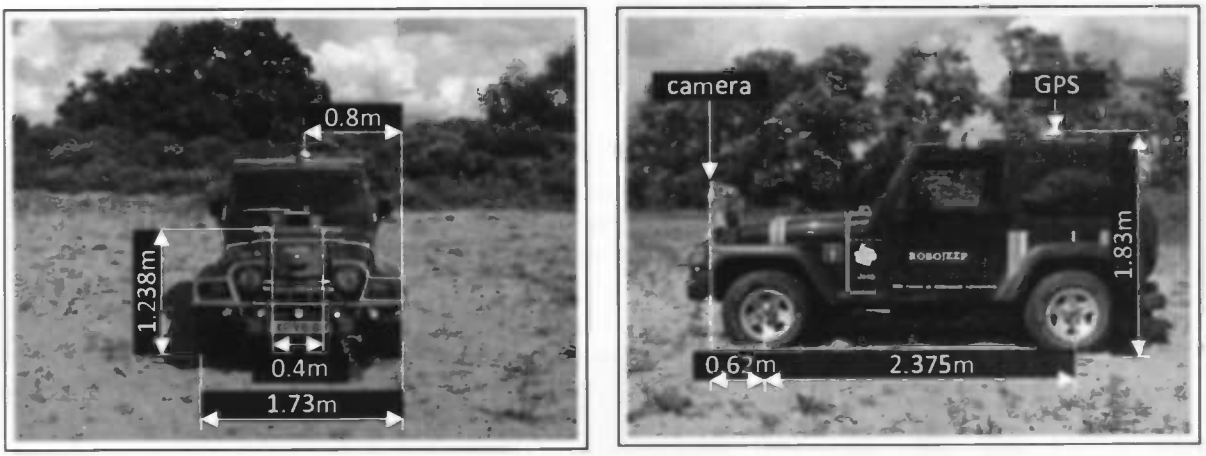

FIGURE 9-1, DIMENSIONS OF THE TNO ROBOJEEP

The two cameras that are mounted at the front of the Robojeep serve as a stereo camera. The baseline of the stereo camera is 40 centimeter and each camera captures 30 gray scale frames per second (fps), with a resolution of 640 by 480 pixels. The pitch of the cameras is a tradeoff; the ground plane and the area above the ground plane must both be captured. Calibration of the cameras is done by using of images of a checkerboard and a calibration toolbox in Matlab, made by Jean-Yves Bouguet [5]. Examples of the images of the checkerboard are shown at the next page.

[5] Camera calibration toolbox: http://www.vision.caltech.edu/bouguetj/calib_doc

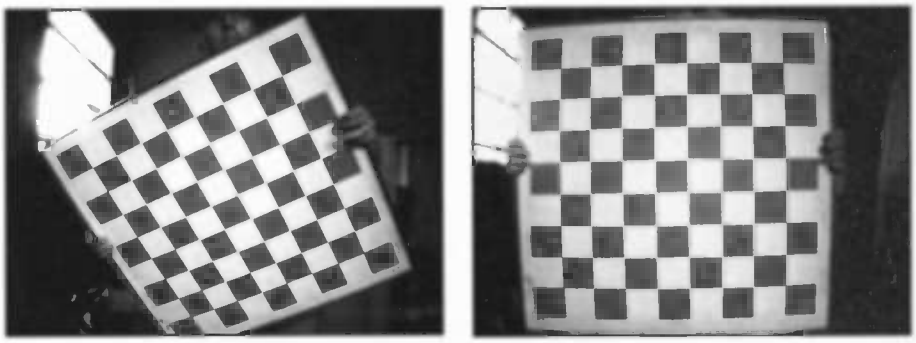

FIGURE 9-2, EXAMPLES OF IMAGES CAPTURED FOR INTERNAL PARAMETER CALIBRATION. THESE PARAMETERS ARE USED TO RECTIFY THE IMAGES

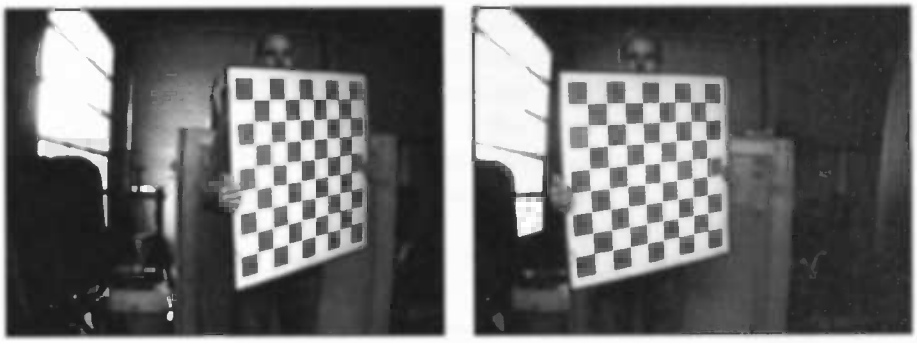

FIGURE 9-3, A LEFT AND RIGHT EXAMPLE OF IMAGES CAPTURED TO CALCULATE EXTERNAL PARAMETERS. THE CHECKERBOARD HAS TO BE PERCEIVED BY BOTH CAMERAS SIMULTANEOUSLY. EXTERNAL PARAMETERS EXPRESS THE POSITION AND ORIENTATION OF ONE CAMERA WITH RESPECT TO THE OTHER.

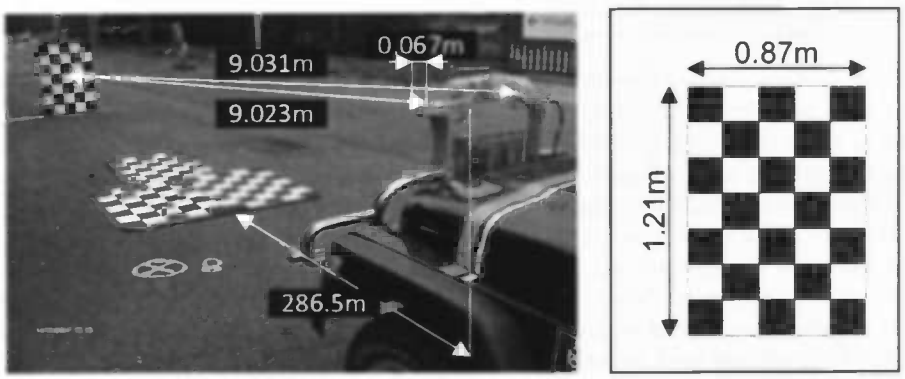

FIGURE 9-4 SHOWS THE SET-UP FOR GROUND PLANE CALIBRATION

FIGURE 9-5, THE LEFT AND RIGHT IMAGE CAPTURED FOR GROUND PLANE CALIBRATION. THIS IS USED TO TRANSFORM BETWEEN THE CAMERA AND WORLD COORDINATES.

When testing the cameras, the sun light was too bright, so a construction with sunglasses was tried. This proved to be ineffective and other cameras were installed. After the first calibration, a data set was captured. When testing the ego-motion estimator, there was a large bias in the noise of the ego-motion estimation, (see also Figure 8-3). To find the cause of the error, a test trajectory was made (see Figure 9-6). Unfortunately, the quality of the captured images was too poor. Because, the bias is highest in the pitch and height of the ego-motion estimation, it was reduced by using data that was captured when driving along a straight road for about 50 meter.

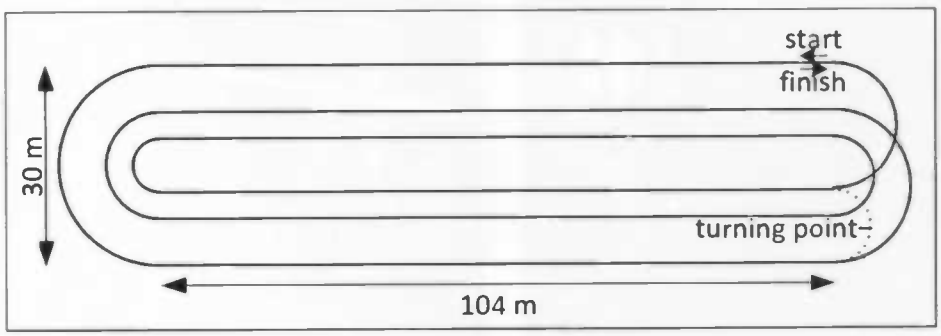

FIGURE 9-6, THE TEST ROUTE ON THE PARKING AREA AT TNO WAS INTENDED TO BE USED TO REDUCE THE NOISE BIAS (FIGURE 8-3). THE SIZE OF THE TEST ROUTE IS MEASURED BY A LASER RANGE FINDER. THE STRAIGHT LINES ARE PERMANENT ON THE PARKING PLACE AND THE CURVES ARE TEMPORARILY LINES DRAWN, WITH CHALK. THE ROUTE CONTAINS SEVERAL CURVES OF DIFFERENT DIAMETERS. AT FIRST, ALL THE CURVES ARE LEFTWARDS THEN, AFTER USING THE TURNING POINT, ALL THE CURVES ARE RIGHTWARDS. THE LINES ON THE GROUND ARE USED AS GROUND PLANE TO REDUCE BIAS IN THE ERROR OF THE ESTIMATOR AT DIFFERENT STEERING ANGLES. UNFORTUNATELY, THE CAPTURED IMAGES ARE TOO DARK TO BE PROCESSED. THIS IS BECAUSE THE ROUTE WAS DRIVEN IN THE EVENING WHEN THE PARKING AREA WAS EMPTY. THE VISIBILITY WAS ALSO REDUCED BECAUSE OF THE HEAVY RAINFALL. LATER ON, THE GROUND PLANE CALIBRATION IS IMPROVED BY DRIVING ALONG A STRAIGHT ROAD FOR ABOUT 50 METER DURING DAYTIME AND USING THE INITIAL EGO-MOTION ESTIMATION TO RE-ADJUST THE GROUND PLANE MATRIX.

The single hard disc in the capturing computer is not fast enough to store the images at 30 frames per second (fps). Using only one hard disc, results in too much dropped frames. Therefore, two high-speed hard disks are bought and installed in the computer, each storing the left or the right images. Additionally to the stereo camera, a camcorder is placed on top of the Robojeep to capture footage that is edited into a promo movie on change detection. A laptop is used to store the information of the GPS, the odometry and the IMU. The laptop and the image-capturing desktop are synchronized via a network.

The power supply in the Robojeep is divided in three systems. System 1, is the normal electrical system of a car. System 2 supplies the embedded hardware. System 3 is an extra power supply, which can be used for testing purposes. During the experiments, it is used to power the desktop computer that is temporarily installed in the Robojeep, for images capturing. The convertor of system 3 does not deliver a nice sinusoid when too much power is demanded. Fortunately, the converter of the computer could handle this input, although a penetrating sound came out of it. A problem is the capacity of the batteries of system 3. The batteries last only 15 minutes before they need to be recharged. The uptime is expanded to 30 minutes, by installing an uninterruptable power supply (UPS) that automatically takes over when the power supply of the Robojeep is empty. An even better solution would be a generator but this is not available at TNO at this moment.

The estimated route is compared to the GPS readings, landmarks on the ground and aerial maps. The raw GPS values are in UTM-WGS84-coördinates and are converted to *Rijksdriehoek (RD)* coordinates by a Matlab function, before used as reference data. This function is an implementation of the transformation written by the engineer F.H. Schreutelkamp, available at the site of the observatory *De Koepel*[6]. The RD transformation can be used for GPS readings in The Netherlands and it transforms these readings into coordinates on a flat surface (Figure 9-7). The *X*-axis points from the west to the east. The perpendicular *Y*-axis points from south to north (Figure 9-8).

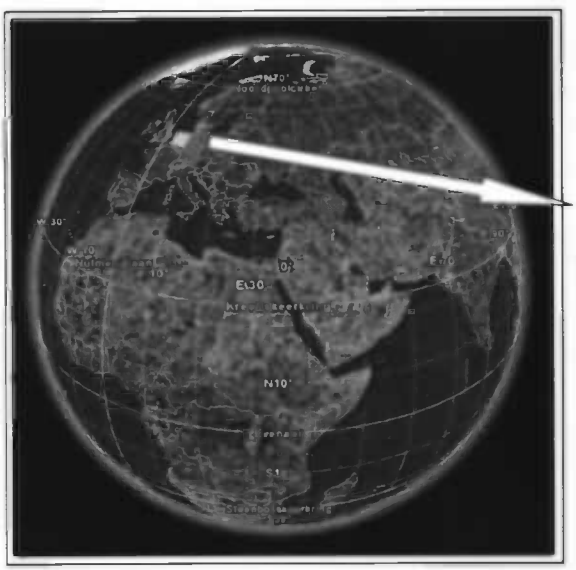

FIGURE 9-7, THE GLOBE WITH A FLAT SURFACE DRAWN OVER THE NETHERLANDS[7]

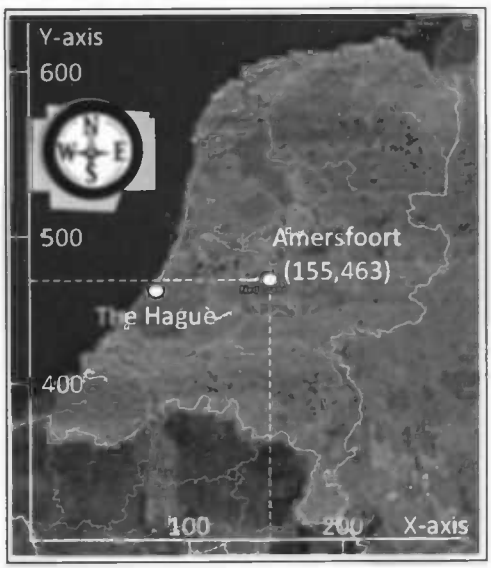

FIGURE 9-8, RIJKSDRIEKHOEK GRID[8])

[6] Observatory De Koepel: http://www.dekoepel.nl/pdf/Transformatieformules.pdf
[7] Google earth: http://earth.google.nl/
[8] Kadaster: http://rdinfo.kadaster.nl/pdf/rd_brochure.pdf

The values of the RD-coordinates are in meters, just like the values of UTM-WGS84 but the central point of the UTM-WGS84 coordinate system is not in the middle of the Netherlands, unlike the central point of the RD system, which results in a projection error because of the spherical shape of the earth (see Figure 9-9).

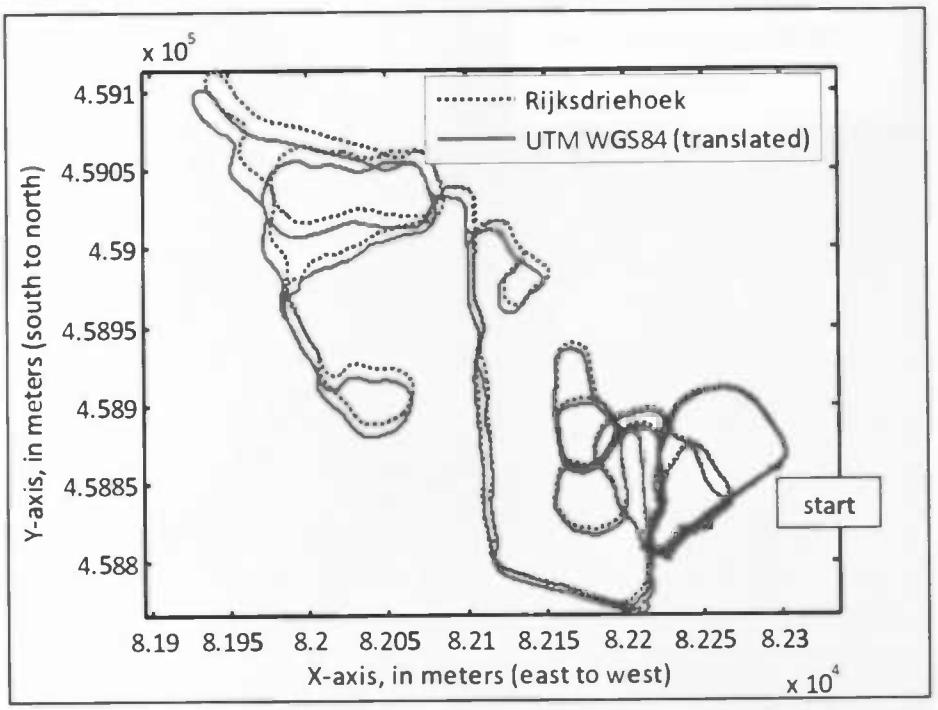

FIGURE 9-9, THE GPS VALUES IN *RIJKSDRIEHOEK* COORDINATES (DOTTED LINE) PLOTTED TOGETHER WITH THE GPS VALUES IN THE ORIGINAL UTM-WGS84 COORDINATES (SOLID LINE) WHERE THE LATTER IS TRANSLATED, SO THE START OF BOTH TRACKS OVERLAP (THE SCALE IS NOT ADJUSTED). THE LINES OVERLAP MORE OR LESS AT THE START BUT THE DEVIATION IS CLEAR IN THE UPPER LEFT CORNER.

10 RESULTS

The GPS data acquired during the test routes, is shown in Figure 10-1 and Figure 10-2. Not all the data can be examined thorough, because the calculation is very time consuming.

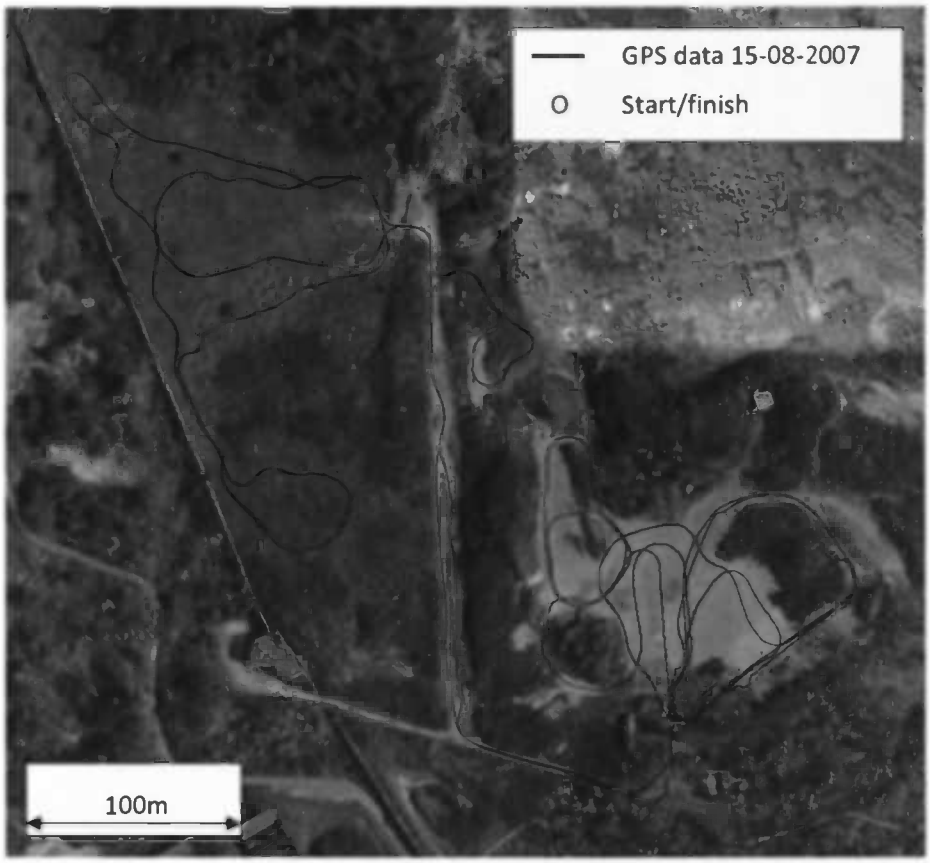

FIGURE 10-1, GPS-DATA OF THE ROUTE AT THE WAALSDORPERVLAKTE (FROM GOOGLE MAPS[9])
THE ROUTE HAS SEVERAL LOOPS IN IT TO TEST LOOP CLOSURE AND REDUCE UNCERTAINTY

[9] Google maps: http://maps.google.nl/

FIGURE 10-2, GPS-DATA OF THE CITY ROUTE IN THE HAGUE (FROM GOOGLE MAPS[10])

[10] Google maps: http://maps.google.nl/

10.1 Loop closure

Figure 10-3, shows the trajectory at which loop closure was tested. A rectified image captured at the start is shown in Figure 10-4. The environment is not completely static. The amount of sunshine varies strong because the sun is shining bright and from time to time a cloud dimmed the light. These changes make the loop closure more difficult. Figure 10-5 shows the sun reflecting on the wet road. This makes it also hard to extract stable SIFT features.

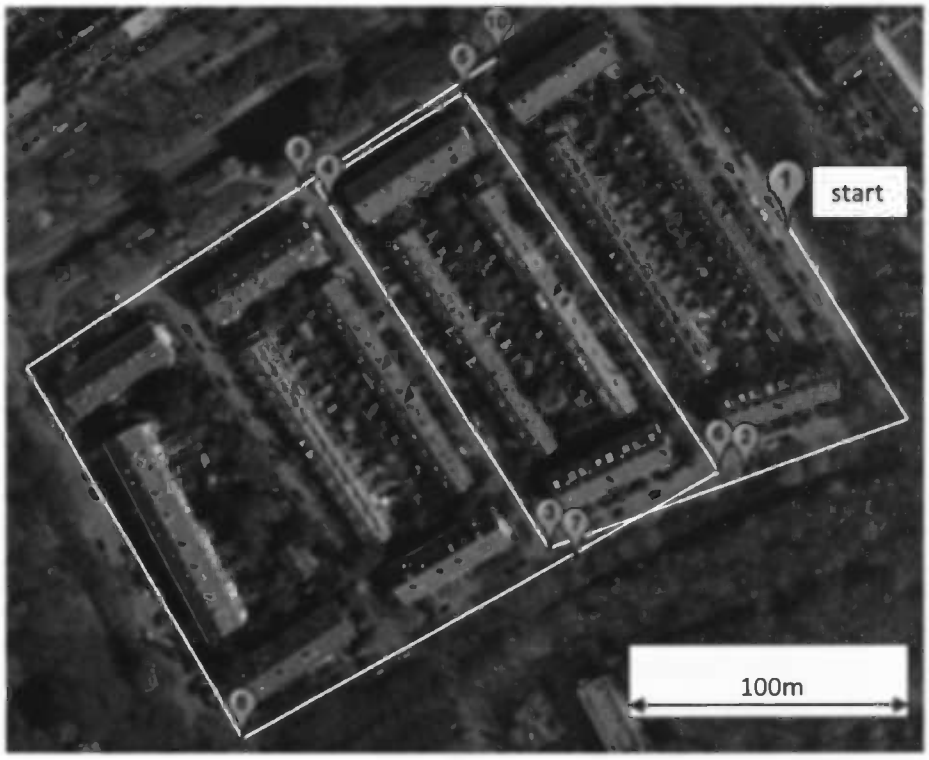

Figure 10-3 shows the part of the city trip that is used for testing loop closure.

Figure 10-4, showing a rectified image captured at the start

Figure 10-5, frame 460, showing the reflection of sun on the wet road. The bright spots make it harder to calculate stable SIFT features.

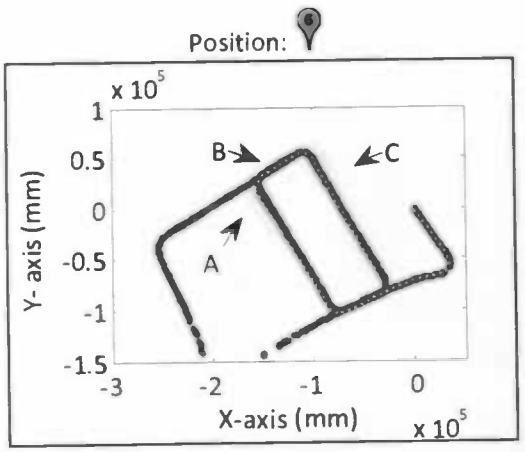

FIGURE 10-6, THE PARTICLE DISTRIBUTION AT TIME STEP 2525. THIS IS BEFORE LOOP CLOSURE TAKES PLACE. THE SPREAD AT ARROW A LOOKS MUCH LARGER THEN AT ARROW B, BUT THE DIFFERENCE IS NOT AS LARGE AS IT SEEMS. DUE TO THE LENGTH OF THE PATH NEAR ARROW A THE SMALL DEVIATION IN THE ANGLE RESULTS IN A LARGE SPREAD. NEAR B THE POSITIONS LOOK ALIKE, BUT THE SPREAD IS LARGE AS CAN BE SEEN AFTER THE NEXT CORNER, WHEN THE PARTICLES ARRIVE AT ARROW C (THE LEGEND IS SHOWN IN FIGURE 10-8).

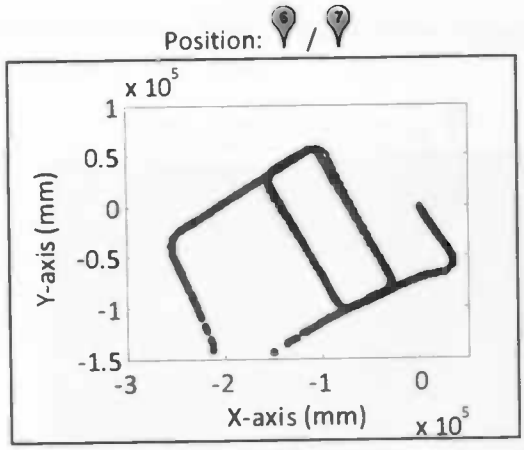

FIGURE 10-7, THE PARTICLE DISTRIBUTION AT TIME STEP 2795. THE THICK LINE SHOWS THE BEST PARTICLE DURING LOOP CLOSURE DETECTION. (THE LEGEND IS SHOWN IN FIGURE 10-8)

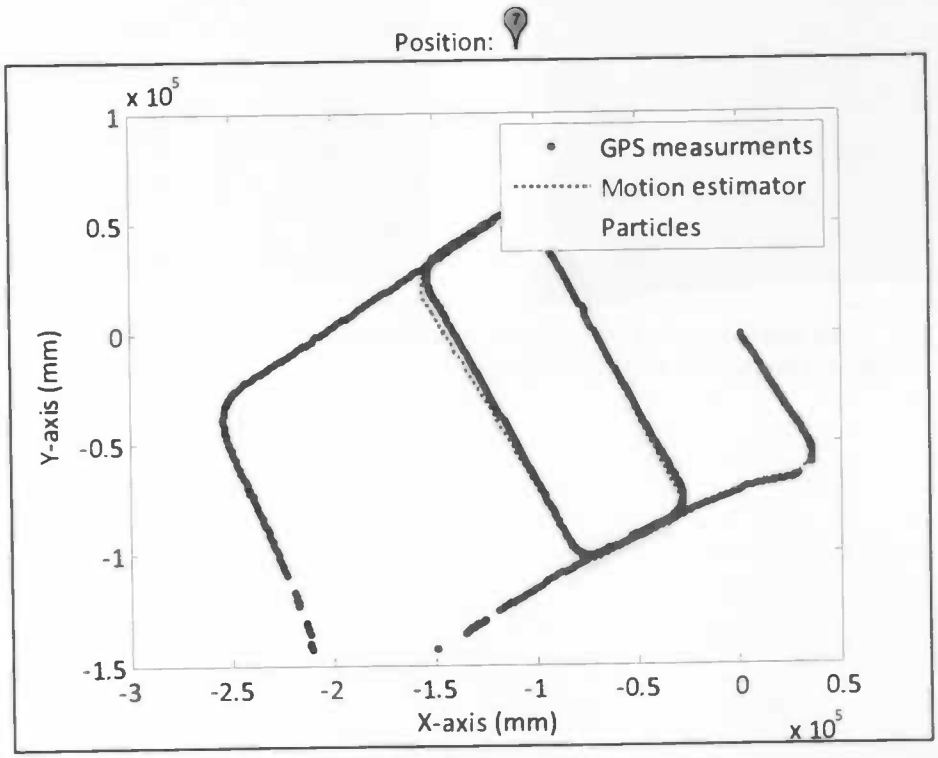

FIGURE 10-8, THE PARTICLE DISTRIBUTION AT TIME STEP 3030. ALTHOUGH THE PARTICLES ARE LESS DISTRIBUTED, BECAUSE OF THE RE-SAMPLING DURING THE FEW LOOP CLOSURE DETECTIONS, DIFFERENT PARTICLE ROUTES REMAIN.

In the city trip, traffic moves in front of the car. An example is shown below. Despite some dynamic objects, the simplification of assuming the environment being static will remain.

FIGURE 10-9, FRAME 3400 THE ROBOJEEP IS APPROACHING A STATIONARY CAR.

FIGURE 10-10, FRAME 3500. THE ROBOJEEP STOPS BEHIND A PARKED CAR

FIGURE 10-11, FRAME 3600. THE ROBOJEEP DRIVES BEHIND AN ACCELERATING CAR

FIGURE 10-12, FRAME 3700 RIGHT. THE ROBOJEEP IS FOLLOWING A CAR, THROUGH A CORNER.

Re-sampling takes place during loop closure. To detect a loop closure, SIFT features seen at different points in time, are compared. In the algorithm, perceived features are compared to features on the map. To illustrate that loop closure detection based on these images is possible, the SIFT features of two images, captures 3251 time steps apart, are matched (see Figure 10-14 and Figure 10-13).

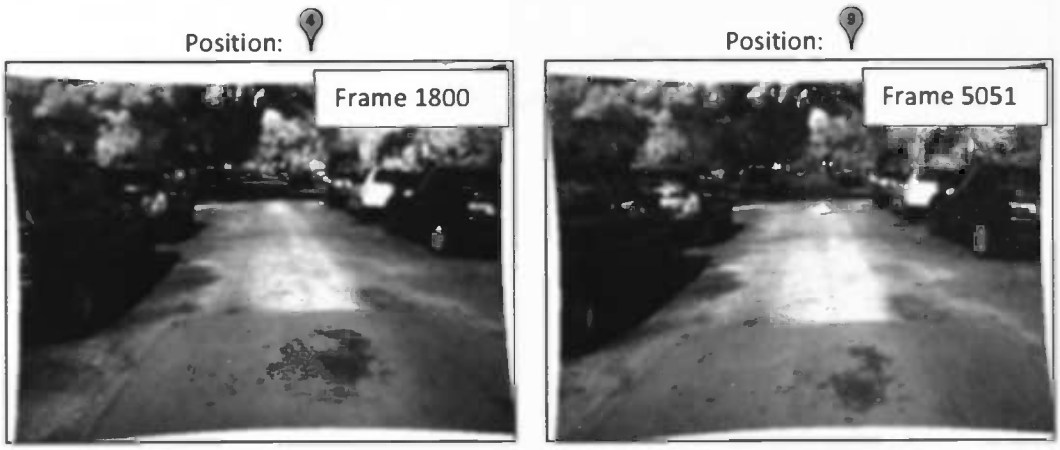

FIGURE 10-13, SHOWS THE IMAGES, CAPTURED BY THE LEFT CAMERA, AT DIFFERENT TIME STEPS

FIGURE 10-14, THE IMAGES OF FIGURE 10-13 ARE SHOWN. THE CIRCLES MARK THE POSITION OF FEATURES THAT HAVE A MATCHING FEATURE IN THE OTHER IMAGE.

Figure 10-15 to Figure 10-18 illustrate the result of re-sampling.

Position:

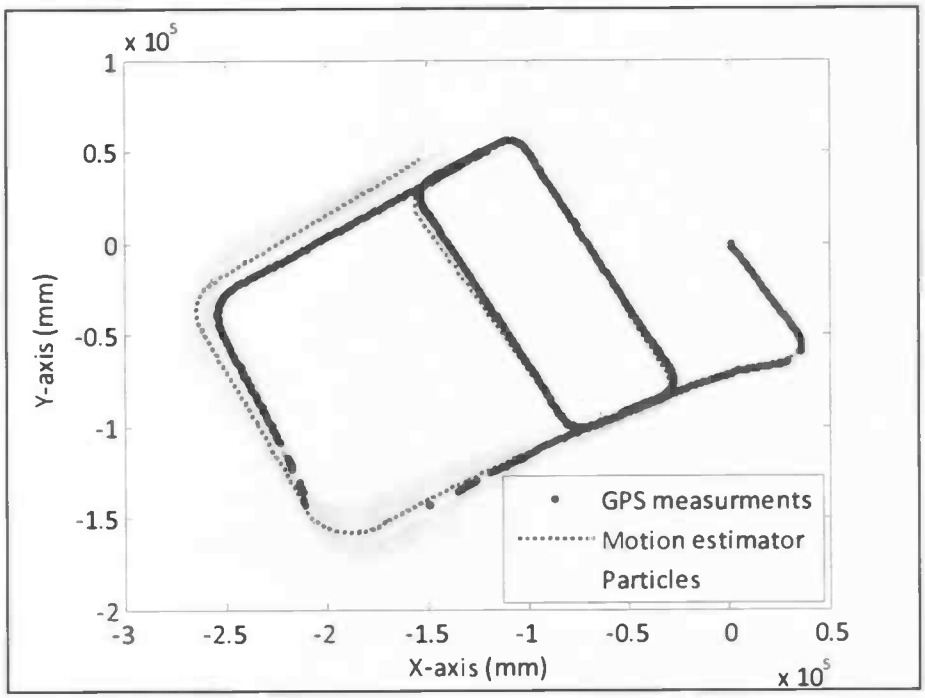

FIGURE 10-15, SHOWS THE PARTICLE DISTRIBUTION AT TIME STEP 4930. THE PARTICLE DISTRIBUTION HAS GROWN AGAIN BECAUSE OF NOISE IN THE EGO-MOTION ESTIMATION

Position:

before

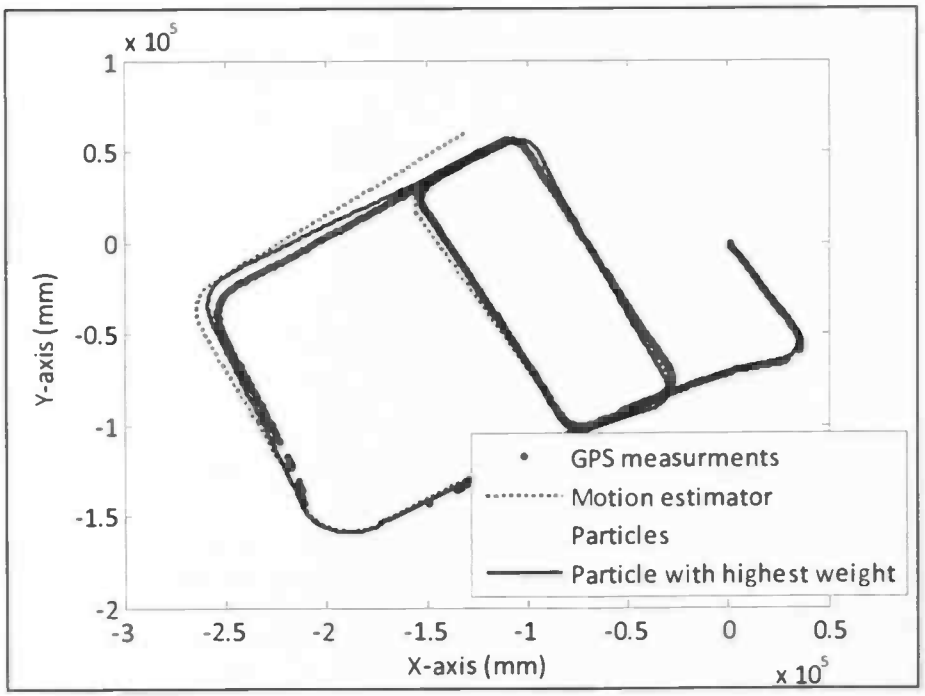

FIGURE 10-16, SHOWS THE PARTICLES, THE RAW MOTION ESTIMATION AND THE GPS MEASUREMENTS. BECAUSE OF LOOP CLOSURE AND RE-SAMPLING HAS TAKEN PLACE MULTIPLE TIMES, THE DISTRIBUTION HAS SHRUNK.

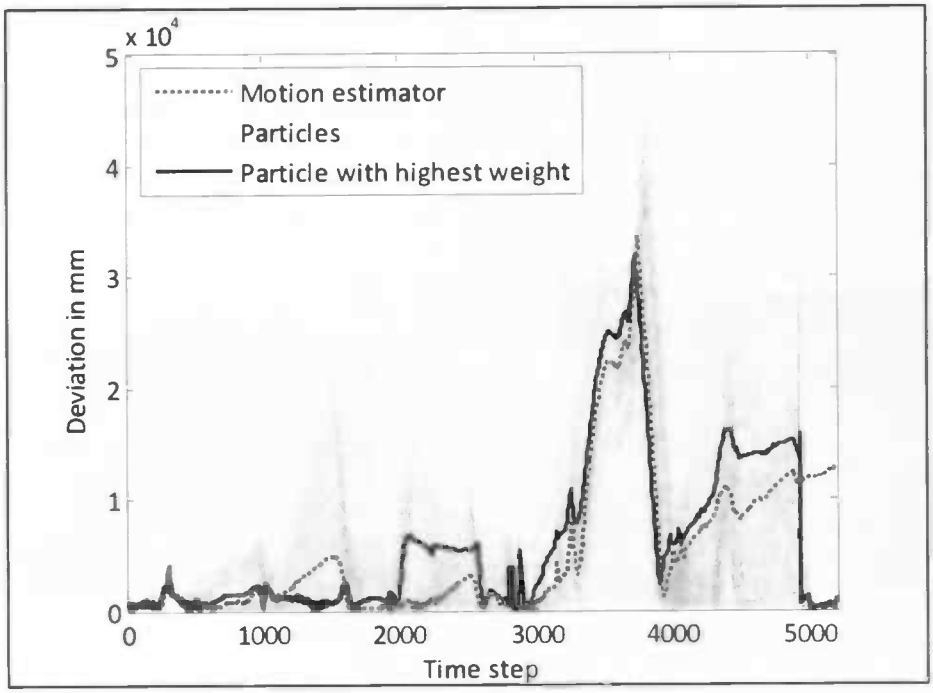

FIGURE 10-17, SHOWS THE DISTANCES TO THE NEAREST GPS MEASUREMENT PER FRAME. IN THE LAST TIME STEPS, WHERE LOOP CLOSURE TAKES PLACE, THE PARTICLES (SOLID GRAY AND BLACK LINES) ARE A FAR BETTER GUESS THEN THE EGO-MOTION ESTIMATOR (DOTTED RED LINE).

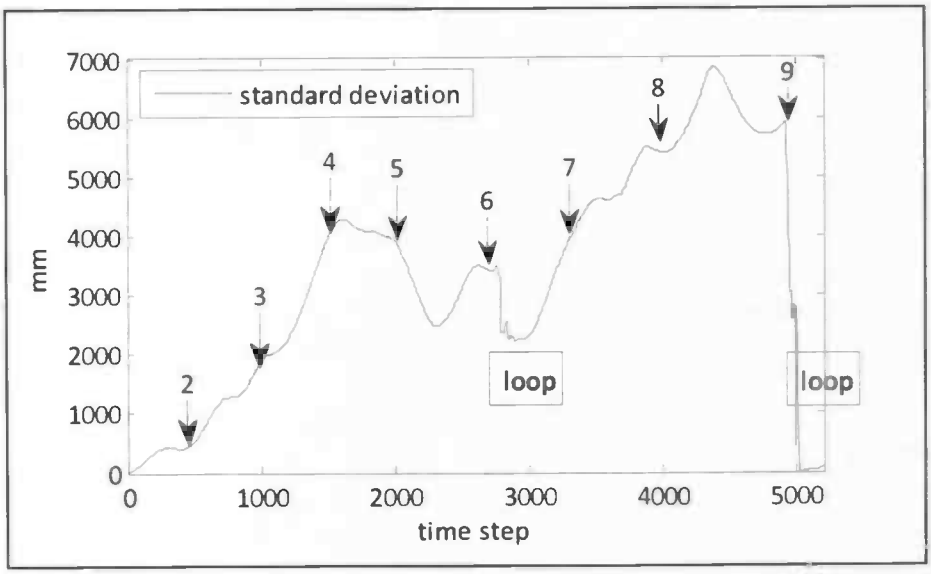

FIGURE 10-18, THE STANDARD DEVIATION OF THE PARTICLE DISTRIBUTION. THE GROWTH OF THE DEVIATION IS NOT CONSTANT BECAUSE OF THE CURVES IN THE ROUTE. THE DEVIATION CAN EVEN DECREASE AFTER A TURN OF 180 DEGREES. AFTER A LOOP CLOSURE THE DEVIATION IS DECREASED FAST.

10.2 PLANAR VERSUS 3D ROBOT MOTION

The environment is not completely static. The wind blows hard, during the test run on the Waalsdorpervlakte, making the long grass and the branches and leaves of the bushes and the trees move back and forth. Quick changes make the ego-motion estimation more difficult. Below, rectified images of the small circular path at the Waalsdorpervlakte are shown. Some captured frames are not used because the time difference between the left and closest right frame is too big.

FIGURE 10-19, THE FIRST LEFT FRAME

FIGURE 10-20, FRAME 600, SHOWING A SMALL HILL

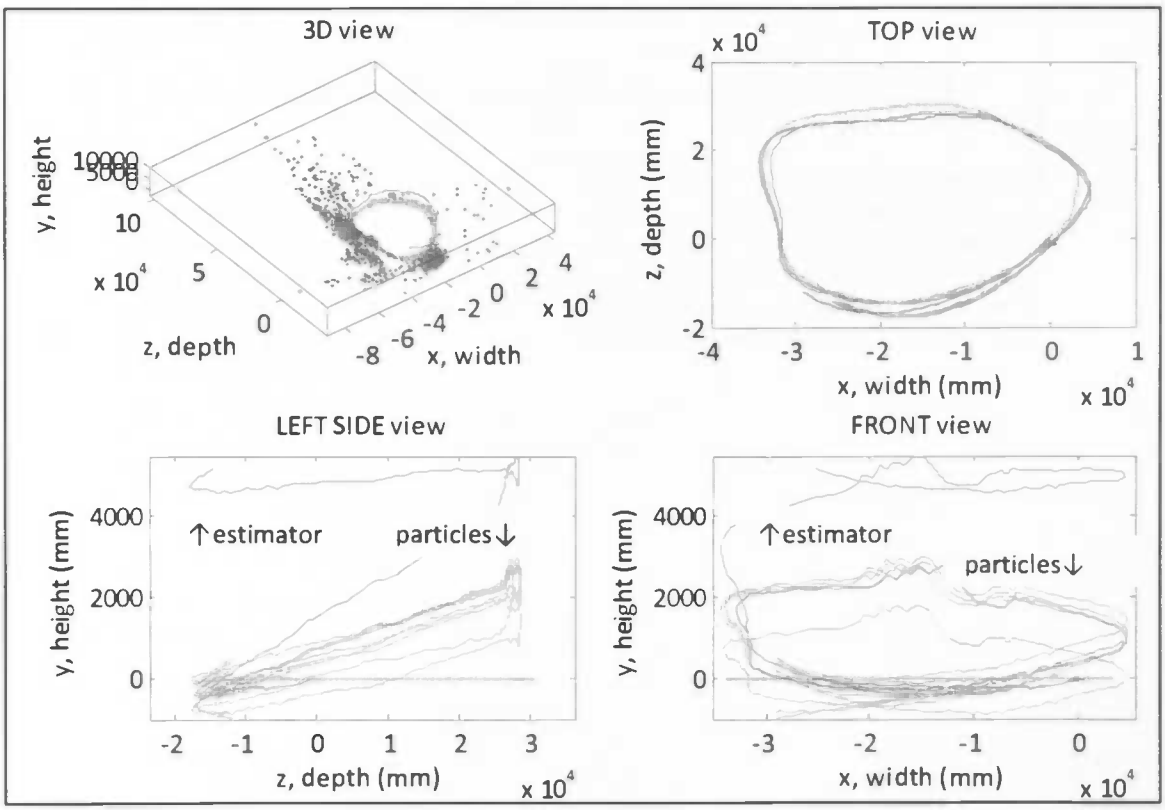

FIGURE 10-21, 3D MOTION. AS CAN BE SEEN IN THE LOWER IMAGES, THE EGO-MOTION ESTIMATOR HAS A BIASED ERROR IN THE PITCH AND/OR HEIGHT ESTIMATION. HERE, THE HORIZONTAL BLUE LINE REPRESENTS THE GPS-VALUES.

10.3 DELETING UNCERTAIN LANDMARKS

Landmarks on and above the ground plane

For the small circular route, the amount of landmarks on the map is plotted in a graph (see Figure 10-22).

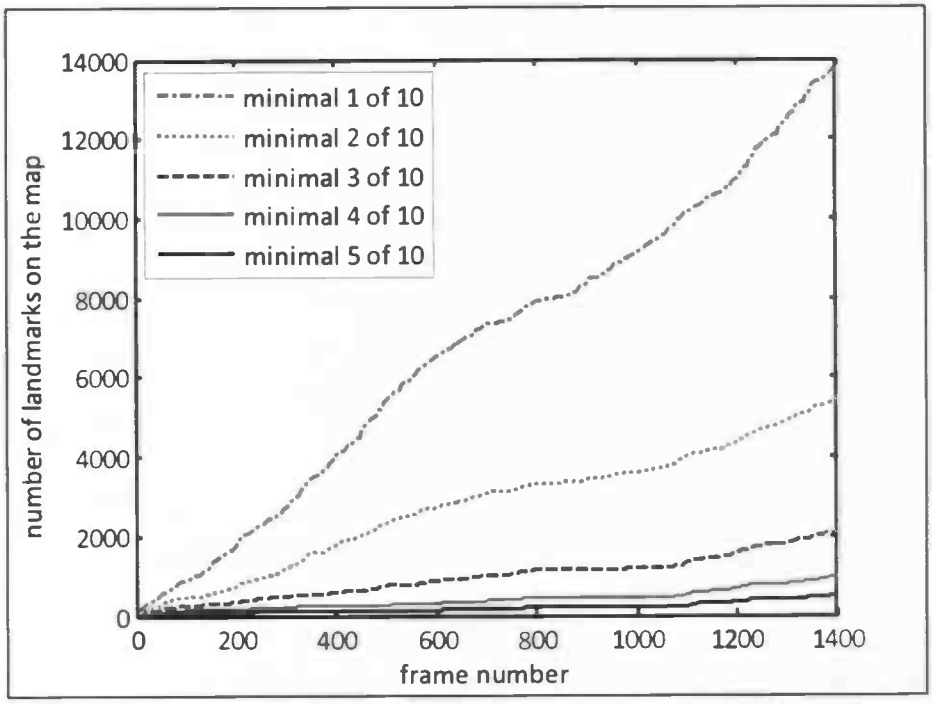

FIGURE 10-22, THE MAP SIZE, IN NUMBER OF LANDMARKS, WHERE THE FRAME STEP SIZE IS 5 (SO EACH 1 IN 5 FRAMES IS PROCESSED). A LANDMARKS IS ADDED TO THE MAP IF THE NUMBER OF OCCURRENCE OF THAT LANDMARK WITHIN 10 PROCESSED FRAMES AFTER THE FIRST OCCURRENCE IS MINIMAL X. THE MAP SIZE IS PLOTTED FOR THE X-VALUES; 1,2,3,4 AND 5. FOR THE X-VALUES 6,7,8,9 AND 10 THE MAP SIZE AT FRAME NUMBER WOULD RESPECTIVELY BE: 247, 129, 64, 32 AND 12.

Landmark above the ground plane

To reduce the number of landmarks further, the landmarks on the ground plane are removed from the dataset that is used for mapping. Landmarks perceived below minus half a meter in height ($Y < -500$) at a distance smaller than 25 meter ($Z < 25000$) are deleted. This result is shown in the graph below (Figure 10-23)

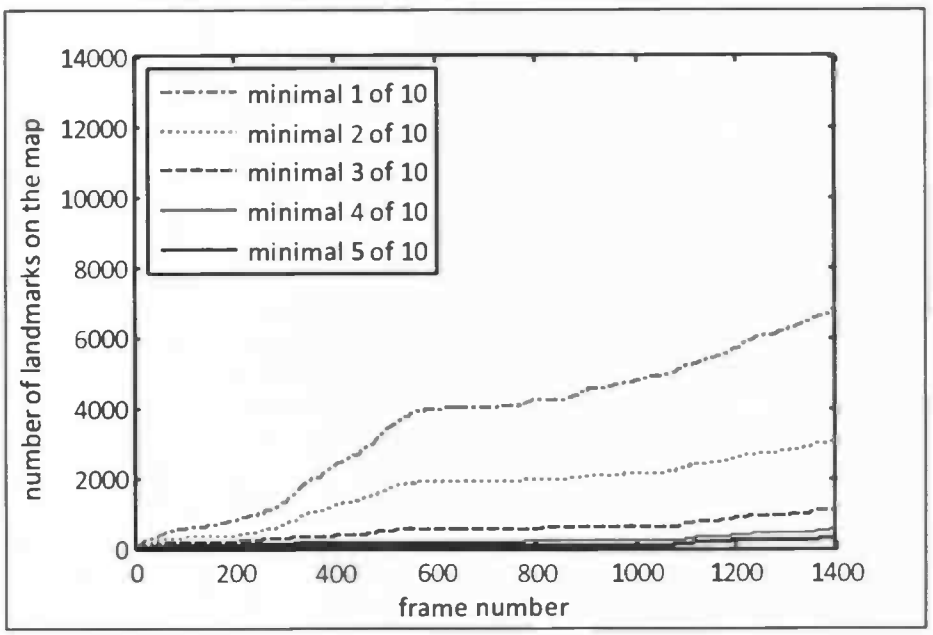

FIGURE 10-23, MAP SIZE IN LANDMARKS, ABOUT THE SAME AS FIGURE 10-22 BUT LANDMARKS THAT SATISFY $((Y < -500)\ AND\ (Z < 25.000))$ ARE DELETED FROM THE INPUT.

10.4 ABSOLUTE VERSUS RELATIVE MATCHING DISTANCE

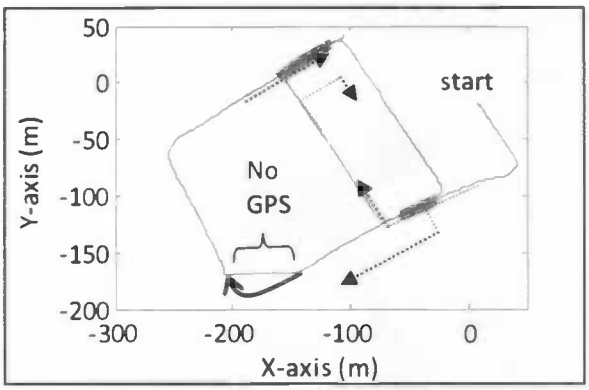

FIGURE 10-24, THE GPS-VALUES OF PART OF THE CITY TRIP (DOTTED BLUE LINE). DURING PART OF THE ROUTE (SEE THE OPEN ARROW AT THE LOWER SECTION OF THE FIGURE), NO GPS SIGNAL WAS RECEIVED BECAUSE OF THE LARGE TREES NEAR THE ROAD. THE GREEN SHADED THICK LINES CORRESPOND TO THE GREEN SHADED THICK LINES IN FIGURE 10-25. THE ROBOT COMES TWICE ACROSS THESE AREAS.

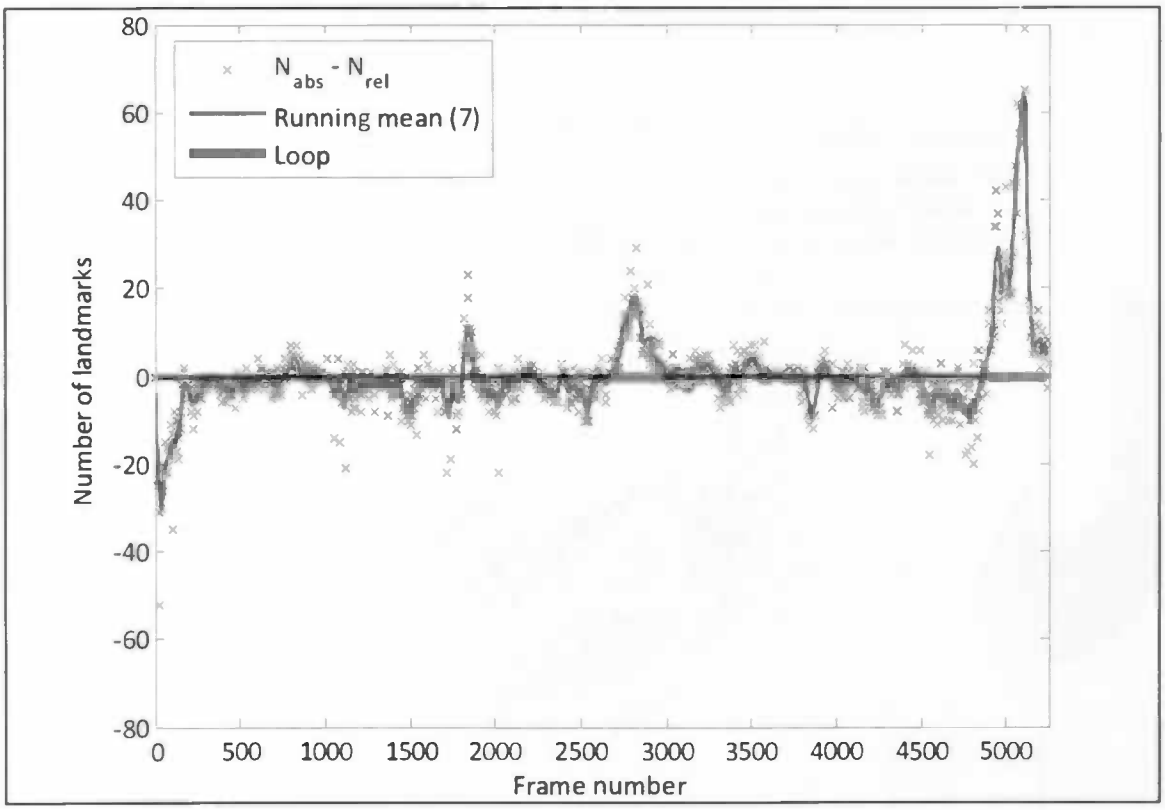

FIGURE 10-25, THE DIFFERENCE BETWEEN THE NUMBER OF MATCHING LANDMARKS, USING TWO DIFFERENT DISTANCE MEASUREMENTS. IN THE GRAPH: $HEIGHT = N_{rel} - N_{abs}$ WHERE: N_{rel} IS THE NUMBER OF MATCHES USING AN ABSOLUTE DISTANCE THRESHOLD OF 0.25 AND N_{abs} IS THE NUMBER OF MATCHES USING A RELATIVE DISTANCE THRESHOLD OF $0.6 * distance_{runner\ up}$. THE DIFFERENCE BETWEEN THE NUMBER OF ABSOLUTE AND RELATIVE MATCHES CAN ALSO BE USED TO DETECT A LOOP CLOSURE USING ONLY SHORT TERM MATCHING

10.5 PRE-SELECTING LANDMARKS FOR MATCHING

The result of preselecting landmarks by their location on the map is illustrated by the two figures below.

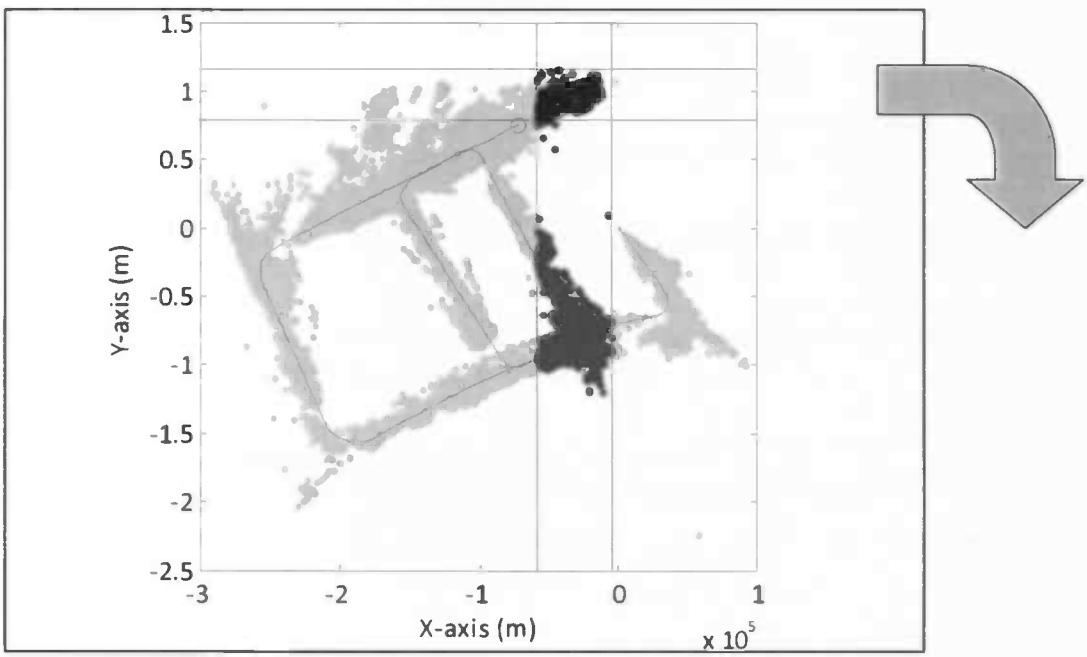

FIGURE 10-26, SELECTING LANDMARKS ON THE MAP FOR MATCHING, FRAME 5000. -=ROUTE OF THE PARTICLE, O=PARTICLE POSITION, •= LANDMARK ON MAP, *=NEW LANDMARK. THE TOTAL MAP HAS 28.743 LANDMARKS AND THE AREA BETWEEN THE STRIPED BLUE LINES HAS 4381 LANDMARKS. THIS IS AFTER SELECTION CRITERIA 1A.

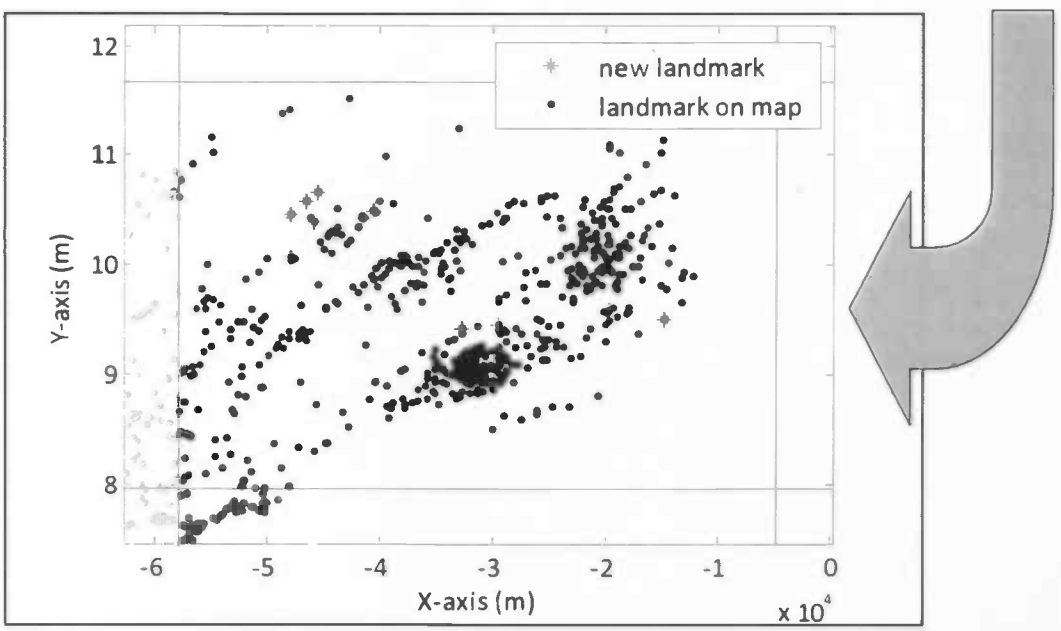

FIGURE 10-27, THE NEW SEARCH SPACE BETWEEN THE LINES, AFTER SELECTION CRITERIA 1A, HAS 570 LANDMARKS. THIS IS ±2% OF THE TOTAL LANDMARKS

10.6 LANDMARK DESCRIPTOR STORED LOCAL OR GLOBAL

Both the storage methods are not tested on the same data set. The local storage is used in the first part of the implementation trajectory, when the data set was relatively small. To give an impression:

- Particle pool: 100 particles
- The amount of frames used is 2000 (processed all).
- The total number of landmarks on each map after the test run is approximately 3000. No other program is allowed to use some of the computers 1 GB internal memory, otherwise the computer will return a memory overload error.

This will take more than 14 hours (also because of memory swapping between RAM and hard disk).

The global storage combined with some local storage is used in the last part of the implementation trajectory on the data of the city trip, after trials with local storage failed.

- Particle pool: 100 particles
- The amount of frames is 5000, which are processed 1 in 5 frames.
- The total number of landmarks on the map after the test run is approximately 28.000.

This will take about 1.5 hours (the second core of the dual core CPU can be used for other programs).

11 CONCLUSION

11.1 LOOP CLOSURE

The loop closure of the implemented SLAM algorithm works as expected (see Figure 10-17). During the end of the route, the particles are within 1 meter from the GPS and the ego-motion estimator is about 12 meters from the GPS. This makes it possible to perform the other experiments. The results are shown in the previous chapter and discussed here.

11.2 PLANAR VERSUS 3D MOTION

The error of the estimator, during the run, is largest in the height (Y-axis). The error is corrected by the SLAM algorithm for a large amount. The improvement due to the SLAM algorithm is therefore relatively high compared to the improvement due to the SLAM algorithm in the planar case (experiment 1). Investigating the motion estimator further reveals that there is a biased error in the pitch estimation. The rest of the experiments are performed by constant setting the estimated pitch moved and changes in height to zero. The route of the robot is therefore planar, but the landmarks are still positioned in three dimensions. From experiment 1 and 2, can be concluded that is it possible to use vision only SLAM in a large-scale environment.

11.3 DELETING UNCERTAIN LANDMARKS

Influence of ground plane landmarks

The results of the third experiment show that leaving out ground plane landmarks reduces the landmarks on the map by about 50 percent. The experiments also show that the loop closure performance does not degrade. However, if the noise level is set high, errors during loop closure can occur when the ground plane landmarks are not used for SLAM. In some situations, a particle with a small positional error and the right angle can be worse, than a particle with a large positional and a large angular error (Figure 11-1).

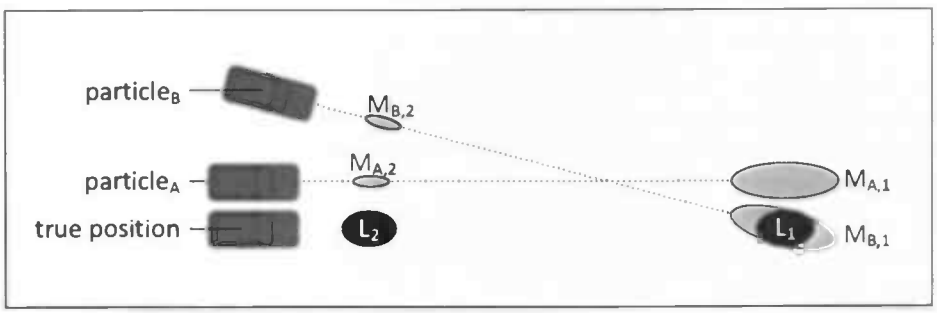

FIGURE 11-1, INFLUENCE OF GROUND PLANE LANDMARK. L_1 IS A LANDMARK ABOVE THE GROUND PLANE AND L_2 IS A LANDMARK ON THE GROUND PLANE. IF THE GROUND PLANE FEATURE IS DELETED LANDMARK, L_1 REMAINS. FEATURE 1 IS PERCEIVED AGAIN DURING A LOOP CLOSURE RESULTING IN MEASUREMENT $M_{A,1}$ AND $M_{B,1}$ OF RESPECTIVELY PARTICLE A AND B. ALTHOUGH THE POSITIONAL AND ANGULAR ERROR OF PARTICLE B IS LARGER THAN THE POSITIONAL AND ANGULAR ERROR OF PARTICLE A, PARTICLE B HAS CURRENTLY THE HIGHEST WEIGHT. WHEN THE GROUND PLANE FEATURES ARE INCLUDE MEASUREMENTS, $M_{A,2}$ AND $M_{B,2}$ WILL ALSO CLOSE THE LOOP BY MATCHING TO THE EARLIER PERCEIVED LANDMARK 2. NOW THE WEIGHT OF PARTICLE A WILL BY THE HIGHEST. THE MAHALANOBIS DISTANCE OF $M_{B,2}$ TO L_2 HAS A LOT OF INFLUENCE, BECAUSE THE COVARIANCES OF $M_{B,2}$ AND L_2 ARE SMALL

Landmarks on the ground plane are close to the robot when they are perceived. They cannot be seen from a large distance, which makes them less interesting to be used as landmarks. On the other hand, if landmarks on the ground plane are perceived close to the robot they have small positional covariances, which make them more useful. If the robot perceives a ground plane landmark again during a loop closure it can more precisely determine its position then when using a distant landmark.

In Figure 11-1, the weight of particle B is higher than that of particle A. This will probably change when the robot travels further. The route of particle B will cross the true route and particle B will then lose weight. However, currently it has higher chance to be redrawn and particle A might disappear from the particle pool. If this is the case, the route of particle A cannot be recovered. It is important that the current weight does not have too much influence of the re-sampling. Particles that have high weights at one moment might not have high weight a few time steps later.

On the other hand, if the influence of the weight is set too low, a loop-closure, which is temporary, might not occur long enough to cancel out the bad particles in favor of the better particles. A right balance has to be found for the re-sampling to be fast enough during a loop closure but not too fast to allow large fluctuations in particle weights to have too much influence.

Deleting uncertain landmarks

It would be an option to delete landmarks that are not seen often. Deleting landmarks that are seen only once or twice after 10 time steps after initialization, reduces the number of landmarks on the map during the short test run on the Waalsdorpervlakte by about a factor 5 (see Figure 10-22 and Figure 10-23). This will speed up the process and it will probably reduce the number of wrong matches because uncertain landmarks are thrown away.

Unfortunately, this gives problems similar to the problem mentioned above. In the extreme case, where landmarks never match when they should match, the map will be empty. The mahalanobis distances for that map will always be zero. A negative contribution to the weight of the map, for a landmark that is not present on the map when it is expected to be present, can be a solution.

11.4 Absolute versus relative matching distance

Lowe, who invented the SIFT features, thought of a way to match these SIFT features. To match a currently perceived feature to the features on the map, the distance to each landmark on the map is calculated. These distances are then sorted and the shortest distance is compared to the runner up. If the shortest distance is less than a certain percentage of the runner up distance, the feature to which the distance is smallest is considered a match. By setting the relative matching threshold to 0.6, the short run on the Waalsdorpervlakte had a good result. When using the relative way of matching during the city trip, it proved to be less effective. This is illustrated by the following example.

Example; relative versus absolute matching

First, a relative matching distance is used. If this is the case, then in many situations the absolute matching distance between landmarks can be large and still be a match. This distance can be large, because probably no second best match is close. However, if the number of landmarks on the map grows, a match will be harder to make, which will result in a new landmark on the map, et cetera.

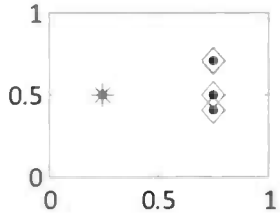

FIGURE 11-2, THE DESCRIPTORS OF TWO LANDMARKS IN A 2D DESCRIPTOR SPACE. THE LANDMARK REPRESENTED BY A DIAMOND IS SEEN THREE TIMES

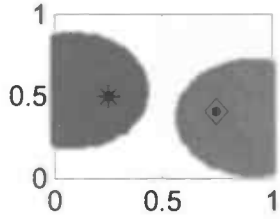

FIGURE 11-3, "RELATIVE" MATCHING AREA OF TWO DESCRIPTORS

$$size\_l_{*} = 23.6\%$$

$$size\_l_{\Diamond} = 23.6\%$$

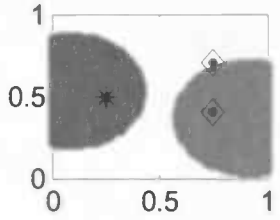

FIGURE 11-4, THE NEW ◊-DESCRIPTOR DOES NOT MATCH

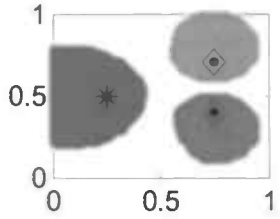

FIGURE 11-5, NEW MATCHING AREAS

$$size\_l_{*} = 20.5\%$$

$$size\_l_{\Diamond} = 12.1 + 11.3 = 23.4\%$$

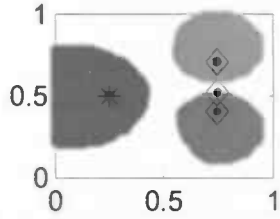

FIGURE 11-6, THE NEW ◊-DESCRIPTOR IS NEAR THE LOWER ◊-DESCRIPTOR BUT DOES NOT MATCH

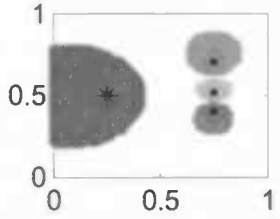

FIGURE 11-7, NEW MATCHING AREAS

$$size\_l_{*} = 0.20\%$$

$$size\_l_{\Diamond} = 5.5 + 1 + 1.4 = 7.9\%$$

IN THE FIGURES, 2 DIMENSIONAL DESCRIPTORS ARE SHOWN (NOT POSITIONS). IN THIS EXAMPLE ONLY TWO DIFFERENT LANDMARKS EXIST, REPRESENTED BY A STAR (*) AND A DIAMOND (◊). IN THE FIRST TIME STEP THE TWO LANDMARKS ARE PERCEIVED (FIGURE 11-2). THERE IS NO MATCH SO THEY ARE ADDED TO THE MATCHING SPACE AS NEW LANDMARKS (FIGURE 11-3). IF THE ◊-LANDMARK IS PERCEIVED AGAIN, ITS DESCRIPTOR HAS SLIGHTLY CHANGED. THIS CHANGE WAS JUST TOO MUCH TO BE A MATCH TO ITS DESCRIPTOR IN THE MATCHING SPACE (FIGURE 11-4) SO AN EXTRA LANDMARK IS ADDED. THIS REDUCES THE TOTAL RELATIVE MATCHING AREA (FIGURE 11-4). IN A LATTER TIME STEP, THE ◊-LANDMARK IS PERCEIVED AGAIN (FIGURE 11-5). ALTHOUGH ITS DESCRIPTOR IS CLOSE TO BOTH THE OTHER ◊-DESCRIPTORS, IT HAS NO MATCH (FIGURE 11-6). THE TOTAL RELATIVE MATCHING AREA HAS NOT ONLY SHRUNK (FIGURE 11-7), THE LANDMARK THAT IS PERCEIVED MOST HAS THE SMALLEST AREA.

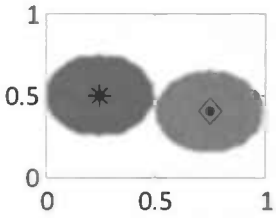

FIGURE 11-8, "ABSOLUTE" MATCHING AREA.

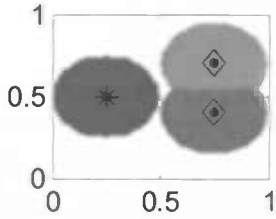

FIGURE 11-9

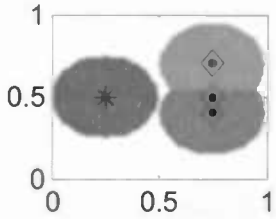

FIGURE 11-10

$size\_l_{\bullet} = 16.2\%$

$size\_l_{\diamond} = 16.2\%$

$size\_l_{\bullet} = 16.2\%$

$size\_l_{\diamond} = 14.2 + 14.2 = 28.4\%$

AGAIN 2 DIMENSIONAL DESCRIPTORS ARE ILLUSTRATED. AT FIRST THE TWO LANDMARKS ARE PERCEIVED AND ADDED TO THE MATCHING SPACE. THE ABSOLUTE MATCHING AREAS ARE SHOWN (FIGURE 11-6). THE ◊-LANDMARK IS PERCEIVED AGAIN BUT DOES NOT MATCH. IT IS ADDED TO THE MATCHING SPACE AND CHANGES THE MATCHING AREAS. THE INDIVIDUAL MATCHING AREAS ARE SMALLER, BUT THE TOTAL ◊-MATCHING AREA HAS GROWN (FIGURE 11-7). THE ◊-LANDMARK IS PERCEIVED AGAIN AND NOW THERE IS A MATCH (FIGURE 11-8).

The research question to be answered in experiment 3 and 4 is: What kind of landmarks can be used best and how can they be matched? The first conclusion is that it is best to use landmarks above the ground plane, if the uncertainty about the pose of the robot is not too large. The second conclusion is that the most uncertain landmarks can be deleted without degrading the performance of the loop closure. The third conclusion is that an absolute matching distance is better than a relative matching distance.

11.5 PRE-SELECTING LANDMARKS FOR MATCHING

On the map are 28.743 landmarks at the end of the run. After the pre-selection only 570 (2%) have to be match to the new perceived landmarks. Although pre-selections uses some processor time, it is efficient to perform.

11.6 LANDMARK DESCRIPTOR STORED LOCAL OR GLOBAL

Storing the landmarks descriptor, the initialization moment and the score of the landmarks on the map globally reduces the amount of storage needed, drastically. The global storage has even more influence on the time it takes to process the test run. The descriptors of the new perceived landmarks need only be matched once to the descriptors of the landmarks on the global map.

Another major advantage of using global map storage for the descriptors is the following: If local matching is used instead of global matching, matching has to be done for each map. A map where landmarks are not matched (and merged) when in reality they should have matched is more likely to have a higher weight. This is because the landmark on the map that was matched correctly, the positional covariance will decrease; the positional covariance of the landmark on the map and the positional covariance of the new measurement are merged into a smaller covariance. Later on, when the landmark is matched during a loop closure, an equal positional difference result in a larger mahalanobis distance on the map where landmarks were merged compared to the map where the landmarks were not merged.

The conclusion of experiment 5 and 6 is that the efficiency of the algorithm can be increased a lot, by pre-selecting landmarks and by storing part of the landmark information global without degrading loop closure performance.

12 DISCUSSION

In general, most SLAM approaches use accurate laser sensors to measure the position of the landmarks and odometric hardware to derive their motion. This makes the sensor and motion model more accurate. This experiment is fully vision based, using stereo vision and SIFT features

Previous research has been done on SLAM using SIFT (Se, Lowe, & Little, 2002) (Sim, Elinas, Griffin, & Little, 2005). The major difference compared to this experiment, is that the experiments described in those papers are performed indoor. The indoor trajectory, in the paper of Sim et al. (2005) is approximately 67.5m. Our outdoor trajectory is much larger, about 1100 m. This is possible because of the way the landmarks are selected, matched and stored and the total scale is different. The area from which a landmark is visible, is on average much larger outdoor. This makes the features more useful for navigation. Another major difference is the degree of freedom of the movement of the robot. While the outdoor trajectory also consists of dunes, the floor in a building is flat. The authors of the indoor experiment decided to estimate three parameters of the motion, compared to 6 DOF used for the Waalsdorpervlakte test run in this experiment.

The processing time and the memory that is needed for processing the σSLAM algorithm for many landmarks, is large. This is reduced rigorously by storing part of the landmark information on a global map. Another great advantage is that each map has an equal amount of landmarks. With local storage, a map with relative less information has probably a relative high weight. A disadvantage of global storage is that it does not allow exploring multi hypotheses for descriptor matching. Wrong matches are no problem, because in most cases, features that are matched that should not be match, degenerate all the maps evenly. This is a result of cutting off the maximum mahalanobis distance above the maximum innovation distance. Other options to speed up the algorithm are also investigated in this experiment with a positive result. This can be a major step towards real time calculation of visual SLAM.

13 FUTURE WORK

Due to the limited time, only a small amount of experiments could be performed. Some of the ideas that was thought of but not investigated thorough are mentioned below.

To reduce uncertainty the robot can use active vision. It points its camera to the area where it can get the most interesting information. An even more effective way could be active loop closure. In this way, the robot can adjust its route to try to couple an uncertain part of the map to a less uncertain part. To investigate active vision and active loop closure, which would make the Robojeep more autonomous, closure an online implementation of the algorithm is needed.

While the algorithm is still rather slow, the structure of a particle filter is ideal to be implemented in parallel because each particle stores its own local map and most calculation, apart from the re-sampling, are performed locally. A Graphical Processing Unit (GPU), might be used where each unified shader represents a particle. A real-time GPU implementation of the SIFT feature point extraction algorithm is available at OpenVIDIA[11], which is a GPU accelerated Computer Vision Library. See also GP-GPU.org[12]; a site about General-Purpose calculations on a GPU.

By adding an omni vision camera the robot is able to perceive its environment in all directions simultaneously. Stereo vision has the advantage that the depth of the landmarks can be calculated. This makes the motion estimation and the initialization of the landmarks on the map less complex. The omni-camera can be used to track the landmarks, so the changes of the SIFT feature descriptor of the landmark over a larger period can be stored. When the robot perceived the landmark from multiple directions, the robot can approache the landmark from multiple directions to closing a loop.

By using dense stereo (Van der Mark, W.; Gavrila, D. M., 2006) the depth of most pixels in the images can be calculated. Using this information, a 3D-reconstructed image can be rendered for different positions/ angles. The SIFT algorithm can be tested to match the rendered image to the current perceived image. Although, it has been shown that no detector-descriptor combination performs well for viewpoint changes of more that $25/30^0$ (Moreels & Perona, 2005), taking occlusion and changes in viewpoint angles into account, by rendering an 3D image, can have a positive influence on the results.

SIFT features are usable on the short term for estimating the ego-motion. For loop closure however, because of their limited invariance, they are not so ideal. By plotting the 2D pixels in 3D space, the 3D-environment can be reconstructed. Loop closure can benefit from this 3D-reconstruction by fitting the currently observed pixel cluster to it. Fitting can be done by altering the algorithm that was used for the ego-motion estimation. The particles that result a 3D-reconstruction with the best fit to current observation gets the highest weight. To use this technique, a solution has to be found to reduce the number of pixels added to the 3D-model each frame.

The reconstructed 3D-environment can also be used for visualization. An operator can "walk" through this virtual space to get a better idea of the situation. This will benefit the awareness of the operator. For this purpose, the amount of information can be reduced by increasing the time interval for capturing, so fewer frames are used. The rendering of the virtual environment can be speeded up by representing groups of points that lay together in the same plane by a single triangle. This triangle can then be textured by a pre-cashed image. This is illustrated by Figure 13-1 to Figure 13-18.

[11] OpenVIDIA, a GPU accelerated Computer Vision Library: http://openvidia.sourceforge.net

[12] GPGPU, a site for General-Purpose calculations on a GPU: http://www.gpgpu.org

FIGURE 13-1, FRAME 100, RECTIFIED

FIGURE 13-2, DEPTH VALUES (IN METERS)

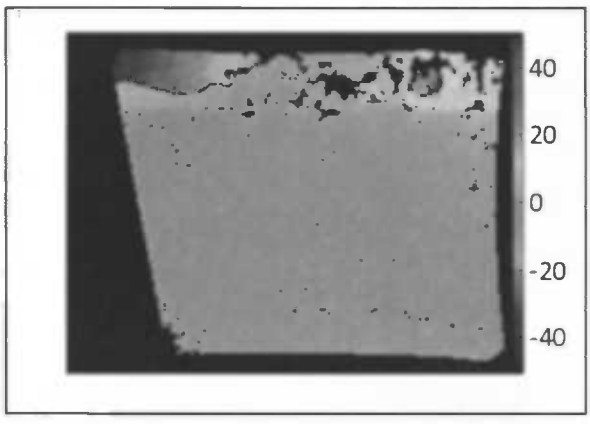

FIGURE 13-3, WIDTH VALUES (IN METERS)

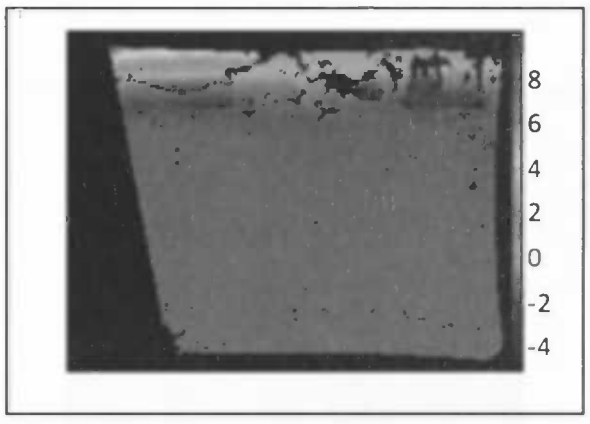

FIGURE 13-4, HEIGHT VALUES (IN METERS)

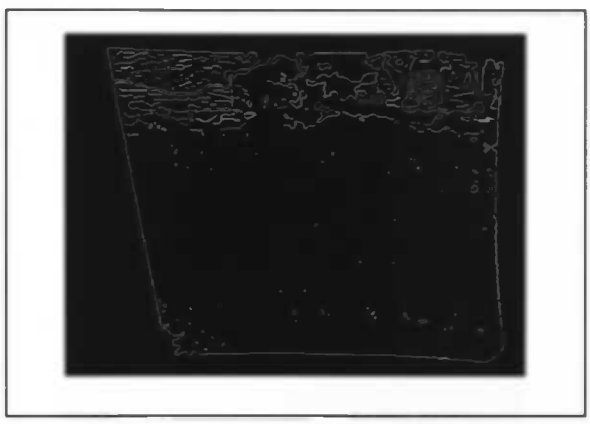

FIGURE 13-5, CANNY EDGE DETECTION ON HEIGHT VALUES

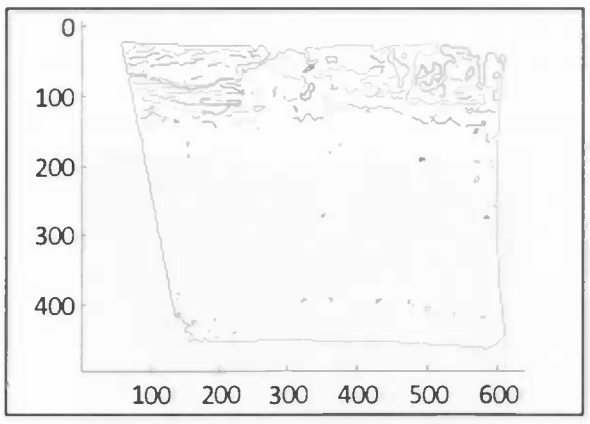

FIGURE 13-6, CONNECTED COMPONENTS (>10 PIXELS)

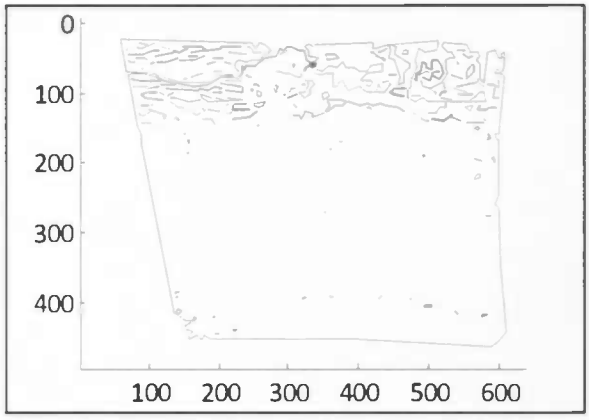

Figure 13-7, Straight line estimate (Deviation max 2 pixels)

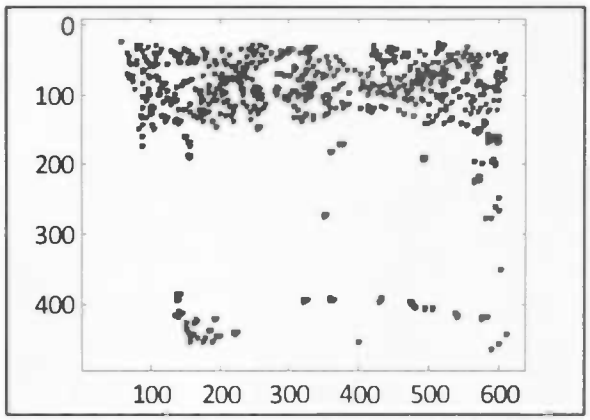

Figure 13-8, Line segment start and end points

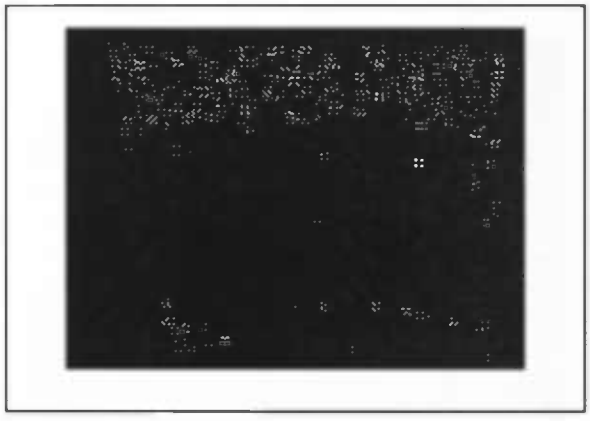

FIGURE 13-9, CONVOLUTION

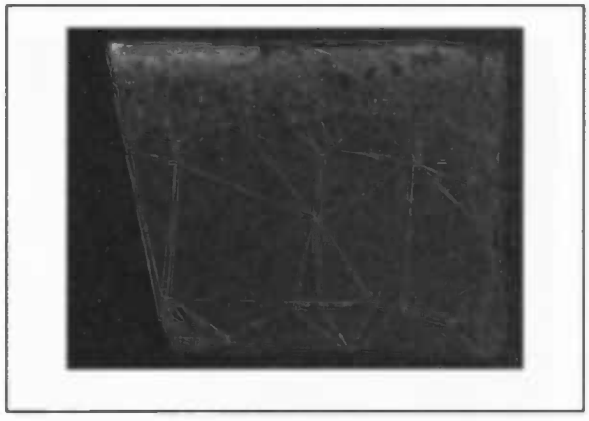

FIGURE 13-10, POINTS CONVERTED INTO TRIANGLES

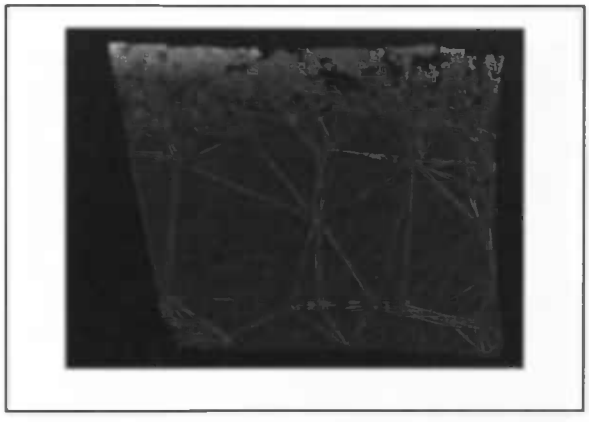

FIGURE 13-11, TRIANGLE LIST, EXCL UNKNOWN AREA'S

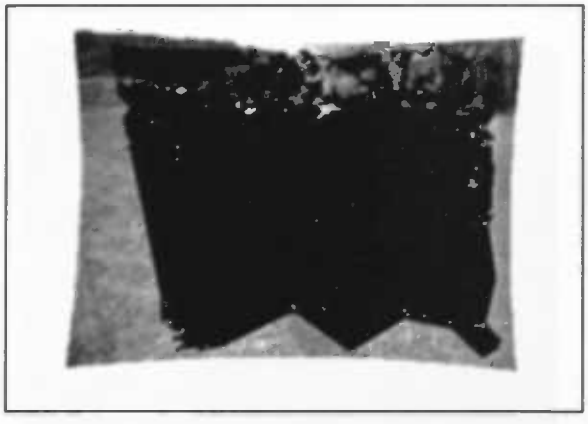

FIGURE 13-12, BLACK AREA'S ILLUSTRATES ARE USED FOR DRAWING

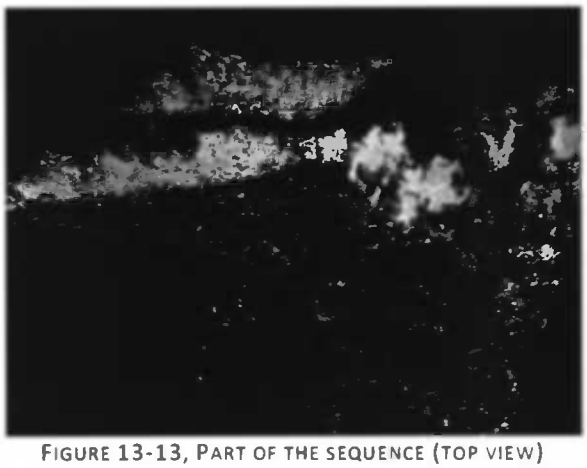

FIGURE 13-13, PART OF THE SEQUENCE (TOP VIEW)

FIGURE 13-14, PART OF THE SEQUENCE (SIDE VIEW)

Pixels of one frame plot in 3D

FIGURE 13-15, ORIGINAL VIEW

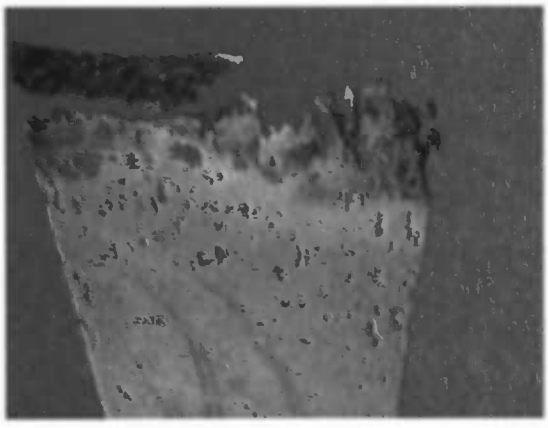

FIGURE 13-16, FROM ABOVE

FIGURE 13-17, FROM LEFT SIDE

FIGURE 13-18, FROM RIGHT SIDE

14 BIBLIOGRAPHY

Arulampalam, S., Maskell, S., Gordon, N., & Clapp, T. (2002). A tutorial on particle filters for on-line non-linear/non-Gaussian Bayesian tracking. *IEEE transactions on signal processing , 50* (2), 174-188.

Ashmore, M., & Barnes, N. (2002). Omni-drive robot motion on curved paths: The fastest path between two points is not a straight-line. *Computer Science* , 225-236.

Bailey, T. (2003). Constrained initialisation for bearing-only SLAM. *International conference on robotics and automation* (pp. 1966-1971). Taipei: ICRA.

Bailey, T. (2002). Experiments in outdoor SLAM. In T. Bailey, *Mobile robot localization and mapping in extensive outdoor environments* (pp. 24-29). Sydney: The university of Sydney.

Bailey, T., Nieto, J., & Nebot, E. (2006, May). Consistency of the Fast-SLAM algorithm. *Robotics and automation* , 424-429.

Bailey, T., Nieto, J., Guivant, J., & Stev, M. (2006). Consistency of the EKF-SLAM algorithm. *International conference on intelligent robots and systems* (pp. 1-7). Beijing: IEEE/ IROS.

Bay, H., Tuytelaars, T., & Van Gaal, L. (2006). SURF: Speeded up robust features. *European conference on computer vision* (pp. 111-125). Graz: ECCV.

Borenstein, J., Everett, H. R., Feng, L., & Wehe, D. (1997). Chapter 1: Sensors for dead reckoning. In J. Borenstein, H. R. Everett, L. Feng, & D. Wehe, *Mobile robot positioning: Sensors and techniques* (pp. 21-26). Michigan: University of Michigan.

Casella, G., & Robert, C. P. (1996). Rao-Blackwellisation of sampling schemes. *Biometrica* (1), 81-94.

Davidson, A., Cid, Y. G., & Kita, N. (2004). Real-time 3D SLAM with wide-angle vision. *Proceedings of IFAC/EURON Symposium on Intelligent Autonomous Vehicles* (pp. 1-9). Lisboa, Portugal,: IFAC/EURON.

Davies, E. R. (2005). The three-dimensional world. In E. R. Davies, *Machine vision* (pp. 445-466). San Francisco: Elsevier Inc.

Diosi, A., & Kleeman, L. (2005). *Scan matching in polar coordinates with application to SLAM.* Victoria: Monash university.

Douc, R., Cappe, O., & Moulines, E. (2005). Comparison of resampling schemes for particle filtering. *Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis* (pp. 64-69). Palaiseau: IEEE, ISPA.

Doucet, A., De Freitas, N., & Gordon, N. (2001). An introduction to sequential Monte Carlo methods. In A. Doucet, N. De Freitas, & N. Gordon, *Statistics for engineering and information science* (pp. 3-14). New York: Springer-Verlag.

Doucet, A., De Freitas, N., Murphy, K. P., & Russell, S. J. (2000). Rao-Blackwellised particle filtering for dynamic Bayesian networks. *Proceedings of the 16th conference on uncertainty in artificial intelligence* (pp. 176-183). San Francisco: Morgan Kaufmann Publishers Inc.

Durrant-Whyte, H., & Bailey, T. (2006). Simultaneous localisation and mapping (SLAM): Part I The essential algorithms. *Robotics and automation magazine* , 99-110.

Eliazar, A., & Parr, R. (2003). DP-SLAM: Fast, robust simulataneous localizationn and mapping without predetermined landmarks. *18th Int. joint conference on artificial intelligence* (pp. 1135-1142). Acupulco: IJCAI-03.

Elinas, P., Sim, R., & Little, J. J. (2006). σSLAM: Stereo vision SLAM using the Rao-Blackwellised particle filter and a novel mixture distribution. *IEEE International conference on robotics and automation (ICRA)* (pp. 1564-1570). Edmonton: IEEE Press.

Fox, D., Hightower, J., Kauz, H., & Liao, L. (2003). Bayesian techniques for location estimation. *Pervasive computing, IEEE , 2* (3), 24-33.

Frese, U. (2006). A discussion of simultaneous localization and mapping. *Autonomous robots* , 25-42.

Hartley, R., & Zisserman, A. (2000). Camera models. In R. Hartley, & A. Zisserman, *Multi view geometry in computer vision* (pp. 153-158). Cambridge: Cambridge university press.

Jung, K., & Rad, A. B. (2001). A robust interest point matching algorithm. *International conference on computer vision* (pp. 538-543). Vancouver: ICCV.

Kalman, R. (1960). A new approach to linear filtering and prediction problems. *Journal of basic engineering , 82*, 35-45.

Kanatani, K. (2005). *Statistical optimization for geometric computation.* Mineola: Dover publications.

Kwok, N. M., & Rad, A. B. (2006). A modified particle filter for localization and mapping. *Intelligent and robotic systems , 46*, 365-382.

Lalonde, M., Byrns, D., Gagnon, L., Teasdale, N., & Laurendeau, D. (2007). Real-time eye blink detection with GPU-based SIFT tracking. *Canadian conference on computer and robot vision* (pp. 481-487). Montreal: IEEE Computer Society.

Lowe, D. G. (2004). Distinctive image features from scale-invariant key points. *International journal of computer vision , 60*, 91-110.

Lowe, D. G. (2001). Local feature view clustering for 3D object recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, (pp. 682-688). Kauai, Hawaii.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. *International conference on computer vision* (pp. 1150-1157). Corfu, Greece: ICCV.

Mikolajczyk, K., & Schmid, C. (2005). A performance evaluation of local descriptors. *Transactions on pattern analysis and machine intelligence , 27* (10), 1615-1630.

Montemerlo, M., & Thrun, S. (2003). Simultaneous localization and mapping with unknown data association using FastSLAM. *International conference on robotics and automation* (pp. 1985-1991). Tapei: ICRA.

Montemerlo, M., Thrun, S., & Koller, D. (2002). Fast-SLAM: A factored solution to the simultaneous localization and mapping problem. *Proceedings of the AAAI national conference on artificial intelligence* (pp. 593-598). Edmonton: NCAI.

Montemerlo, M., Thrun, S., Roller, D., & Wegbreit, B. (2003). FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. *International joint conference on artificial intelligence* (pp. 1151-1156). Acapulco: IJCAI.

Moreels, P., & Perona, P. (2005). Evaluation of features detectors and descriptors based on 3D objects. *International conference on computer vision* (pp. 800-807). Beijing: ICCV.

Murphy, K. P. (1999). Bayesian map learning in dynamic environments. *Advances in neural information processing systems* (pp. 1015-1021). Denver: MIT press.

Ramoni, M., & Sebastiani, P. (1999). Bayesian methods. In M. Berthold, & D. J. Hand, *Intelligent data analysis* (pp. 123-132). New York: Springer.

Ramos, F. T., Nieto, J., & Durrant-Whyte, H. F. (2007). Recognising and modelling landmarks to close loops in outdoor SLAM. *International conference on robotics and automation* (pp. 2036-2041). Rome: ICRA.

Scharstein, D., Szeliski, R., & Zabih, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision* , 7-42.

Se, S., Lowe, D., & Little, J. (2002). Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *The international journal of robotics research , 60*, 735-758.

Sim, E., Elinas, P., Griffin, M., & Little, J. J. (2005). Vision-based SLAM using the Rao-Blackwellised particle filter. *IJCAI workshop reasoning with uncertainty in robotics* (pp. 1-8). Edinburgh: IJCAI.

Thrun, S. (2002). *Robot mapping: A survey.* Pittsburgh: Carnegie Mellon university.

Thrun, S., Burgard, W., & Fox, D. (2005). *Probabilistic robotics* (1 ed.). Cambridge: MIT press.

Van de Craats, J. (2003). 6.2 Lineaire afbeeldingen en matrices. In J. Van de Craats, *Vectoren en matrices* (pp. 97-104). Utrecht: Epsilon.

Van der Mark, W. (2007). Ch1: Introduction, ch2: Stereo vision and ego-motion estimation and ch3: Real-time dense stereo disparity estimation. In W. Van der Mark, *Stereo and colour vision technigues for autonomous vehicle guidance* (pp. 1-85). The Hague: ASCI.

Van der Mark, W.; Gavrila, D. M. (2006). Real-time dense stereo for intelligent vehicles. *Intelligent transport systems , 7* (1), 38-50.

Webb, A. (2002). Chapter 3: Density estimation. In A. Webb, *Statistical pattern recognition* (pp. 81-111). Malvern: John Wiley & sons, LTD.